

THE EXPERT'S VOICE® IN SQL SERVER



Covers
Release 2!

Foundations of SQL Server 2008 R2 Business Intelligence

*Create business intelligence using PowerPivot,
SSAS, SSIS, SSRS, and other BI tools*

SECOND EDITION

Guy Fouché and Lynn Langit

Apress®

For your convenience Apress has placed some of the front matter material after the index. Please use the Bookmarks and Contents at a Glance links to access them.



Apress®

Contents at a Glance

| | |
|--|-------------|
| Contents | v |
| About the Authors | xiii |
| About the Technical Reviewer | xiv |
| Acknowledgments | xv |
| | |
| ■ Chapter 1: What Is Business Intelligence? | 1 |
| ■ Chapter 2: OLAP Modeling Concepts | 25 |
| ■ Chapter 3: Introducing OLAP Modeling with SSAS | 53 |
| ■ Chapter 4: Intermediate OLAP Modeling with SSAS | 83 |
| ■ Chapter 5: Advanced OLAP Modeling with SSAS | 107 |
| ■ Chapter 6: Cube Storage and Aggregation | 131 |
| ■ Chapter 7: Introducing SSIS | 161 |
| ■ Chapter 8: Intermediate SSIS | 187 |
| ■ Chapter 9: Advanced SSIS | 237 |
| ■ Chapter 10: Reporting Tools | 271 |
| ■ Chapter 11: Data Mining with Excel | 301 |
| ■ Chapter 12: Introducing PowerPivot | 329 |
| ■ Chapter 13: Introduction to MDX | 347 |
| ■ Chapter 14: Introduction to Data Mining | 369 |
| ■ Appendix: The HIERARCHYID Datatype | 403 |
| Index | 407 |



What Is Business Intelligence?

This chapter presents a blueprint for understanding the exciting potential of SQL Server 2008 R2's Business Intelligence (BI) technologies to meet your company's crucial business needs. It describes tools, techniques, and high-level implementation concepts for BI.

This chapter covers

- Defining business intelligence
- Understanding BI from an end-user perspective
- Understanding the business problems BI addresses

Just What Is Business Intelligence?

Business intelligence (BI) is defined in many ways. Often, vendors will craft the definition of BI to show their tools in the best possible light. We like to look at BI as an approach to presenting business data in ways that allows executives to query and manipulate that data to gain business insight needed to make the decisions needed to run the day-to-day business in a (hopefully!) profitable manner.

BI data is displayed in a fashion that is appropriate to each type of user; that is, analysts will be able to drill into detailed data; executives will see timely summaries, and middle managers will see data presented at the level of detail that they need to make good business decisions. BI uses cubes, rather than tables, to store information and presents information via reports. The reports can be presented to end users in a variety of formats: Windows applications, web applications, and Microsoft BI client tools, such as Excel, PowerPivot, or SQL Reporting Services.

Figure 1–1 shows a sample of a typical BI physical configuration. You'll note that Figure 1–1 shows a staging database server and a separate BI server. Although it is possible to place all components of BI on a single physical server, the configuration shown in the figure is the most typical for the small-to-medium BI projects that we've worked on. You may also need to include more servers in your project, depending on scalability and availability requirements. You'll learn more about these concepts in Chapter 15.

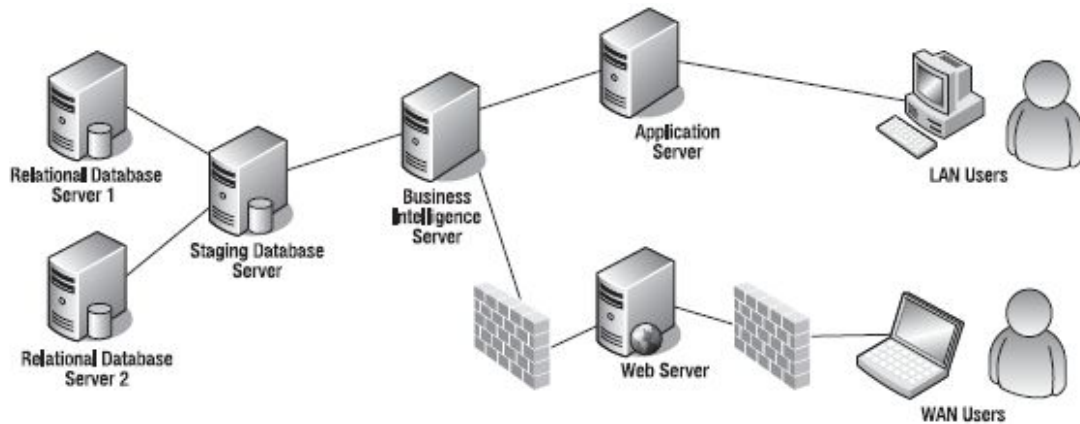


Figure 1-1. An enterprise BI configuration

In addition to the term *business intelligence*, several other terms are commonly used in discussing the technologies depicted in Figure 1-1:

Data warehouse: A data warehouse is an enterprise data repository that houses a single, unified version of business data. Data warehouses are also used to hold an aggregated, or rolled-up and read-only view, of an organization's data; sometimes this structure includes client query tools.

■ **Tip** Data warehousing is not new. The most often-quoted spokespeople from the world of data warehousing theory are Bill Inmon and Ralph Kimball. Both have written many articles and books and have very popular web sites talking about their experiences with data warehousing solutions using products from many vendors.

To read more about Ralph Kimball's ideas on data warehouse design modeling, go to <http://www.ralphkimball.com>. We prefer Kimball's approach to modeling and have had good success implementing Kimball's methods in production BI projects.

Data mart: A defined subset of a data warehouse, a data mart (see Figure 1-2) may represent one business unit (for example, marketing), or a business subject area (for example, loss mitigation) from a greater whole (that is, the entire organization).

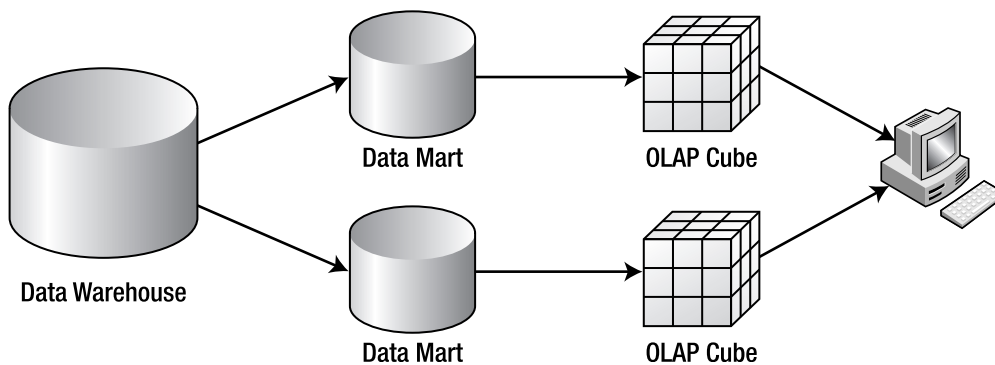


Figure 1–2. Data marts are subsets of enterprise data warehouses and are often defined by time, location, or department.

Cube: A storage structure used by classic data warehousing products in place of many (often normalized) tables. Rather than using tables with rows and columns, cubes use dimensions and measures (or facts). Also, cubes will usually present data that is aggregated (usually summed), rather than presenting each individual item (or row). The structure of a cube is often described this way: cubes present a summarized, aggregated view of enterprise data, as opposed to normalized table sources that present detailed data. Cubes are populated with a read-only copy of source (or production) data. In some cases, cubes contain a complete copy of production data; in other cases, cubes contain subsets of source data. The data is moved from source systems to the destination cubes via extract, transform, and load (ETL) processes. We will discuss cube dimensions and facts in greater detail in Chapter 2.

■ **Note** Another name for a cube or set of cubes is an *online analytical processing* (OLAP) system. When working on BI projects, you may hear the terms *data warehouse*, *cube*, *OLAP*, and *DSS* used interchangeably. Another group of terms you'll hear associated with OLAP are MOLAP, HOLAP, and ROLAP. These terms refer to the method of storing the data and metadata associated with a SSAS cube. The acronyms stand for multidimensional OLAP, hybrid OLAP, or relational OLAP. Storage methods are covered in detail in Chapter 6.

Decision support system (DSS): This term's broad definition can mean anything from a read-only copy of an online transaction processing (OLTP) database to a group of OLAP cubes or even a mixture of both. If the data source consists only of an OLTP database, this store is usually highly normalized. One of the challenges of using an OLTP store as a source for a DSS is the difficulty in writing queries that execute quickly and with little overhead on the source system. This challenge is due to the level of database normalization. The more normalized the OLTP source, the more joins that must be performed on the query. Executing queries that use many joins places significant overhead on the OLTP store. Also, the locking behavior of OLTP databases is such that large read queries can cause significant contention (and thus waiting) for resources by end users. Yet another complexity is the need to properly index the tables in each query. This book is focused on using the more efficient BI store (or OLAP cube) as a source for a DSS system.

NORMALIZATION VS. DENORMALIZATION

What's the difference between normalization and denormalization? Although entire books have been written on the topic, the definitions are really quite simple. *Normalization* means reducing duplicate data by using keys or IDs to relate rows of information from one table to another, for example, customers and their orders. *Denormalization* means the opposite, which is deliberately duplicating data in one or more structures. Normalization improves the efficiency of inserting, updating, or deleting data. The fewer places the data has to be updated, the more efficient the update and the greater the data integrity. Denormalization improves the efficiency of reading or selecting data and reduces the number of tables the data engine has to access or the number of calculations it has to perform to provide information.

Defining BI Using Microsoft's Tools

Microsoft entered the BI market when it released OLAP Services with SQL Server 7.0. It was a quiet entry, and Microsoft didn't gain much traction until its second BI product release, SQL Server 2000 Analysis Services. Since its first market entry, Microsoft has taken the approach that BI should not be for the few (business analysts and possibly executives) but for *everyone* in the organization. This is a key differentiator from competing BI product suites. One implementation of this differentiation is Microsoft's focus on integrating support for SSAS into its Office products—specifically Excel. Excel can be used as a SSAS client at a much lower cost than third-party client tools. The tools and products Microsoft has designed to support BI have been targeted very broadly. In typical Microsoft fashion, they've attempted to broaden the BI usage base with each release.

If you're completely new to BI, it's important for you to consider the possibilities of BI in the widest possible manner when beginning your project. This means planning for the largest possible set of end-user types, that is, analysts, executive managers, middle managers, *and* all other types of end users in your organization. You must consider (and ask your project supporters and subject matter experts [SMEs]) which types of end-user groups need to see what type of information and in what formats (tabular, chart, and so on).

If you have experience with another vendor's BI product (for example, Cognos, Informatica, or Essbase), you may find yourself rethinking some assumptions based on use of those products because Microsoft's BI tools are not copies of anything already on the market. Although some common functionality exists between Microsoft and non-Microsoft BI tools, there is also a large set of functionality that is either completely new or implemented differently than non-Microsoft BI products. This consideration is particularly important if you are migrating to Microsoft's BI from a non-Microsoft BI vendor. We've seen several Microsoft BI production solutions that were needlessly delayed due to lack of understanding of this issue. Whether you are migrating or entirely new to BI, you'll need to start by considering the products and technologies that can be used in a Microsoft BI solution.

What Microsoft Products Are Involved?

At the time of this writing, the most current Microsoft products that support BI (commonly referred to as *the Microsoft BI stack*) are the following:

SQL Server 2008 R2: This is the preferred staging and source location for BI solutions. Data can actually be retrieved from a variety of data stores, such as Oracle, DB2, and Teradata. SQL Server Integration Services (SSIS) is used to perform the ETL of source data into the data warehouse, and most BI solutions will include at least one SQL Server installation. Another key component in many BI solutions is SQL Server Reporting Services (SSRS). When working with SQL Server to perform OLAP administrative tasks, you will use the management interface, which is called SQL Server Management Studio (SSMS).

SQL Server Analysis Services (SSAS): SSAS provides multidimensional storage for the data used in cubes for your data warehouse. SSAS also provides processing and management for those cubes. This product may or may not run on the same physical server as SQL Server 2008 R2. We will detail how to set up cubes in upcoming chapters. Figure 1–3 shows the primary tool—Business Intelligence Development Studio (BIDS)—that you'll use to develop cubes for Analysis Services. You'll note that BIDS opens in a Visual Studio (VS) environment. A full VS installation is *not* required to develop cubes for SSAS. If you do not have VS on your development machine, when you install SSAS, BIDS will install as a stand-alone component. If you do have VS on your development machine, BIDS will install as a component (really a set of templates) into your existing VS instance.



Figure 1–3. You use the Business Intelligence Development Studio (BIDS) to implement BI solutions.

Data mining using SSAS: This component allows you to create data mining structures. These structures include data mining models. *Data mining models* are objects that contain source data (either relational or multidimensional) that have been processed using a particular type of data mining algorithm. These algorithms either classify (group) only or classify and predict one or more column values. Although data mining was available in Analysis Services 2000, Microsoft has significantly enhanced the capabilities of this tool. For example, in the Analysis Services 2000 release there were only two data mining algorithms available, in the 2008 R2 release there are nine data mining algorithms. We will provide an overview of data mining in general and the capabilities available in SSAS for implementing data mining in Chapter 14.

SQL Server Integration Services (SSIS): This toolset is a key component in most BI solutions that is used to import, cleanse, and validate data prior to making the data available to data warehouses, data marts, and SSAS for reporting purposes. It is typical to use data from many disparate sources (relational databases, flat files, XML, and so on) as source data to a data warehouse. For this reason, a sophisticated toolset like SSIS is used to facilitate the complex data loads that are common to BI solutions. As stated earlier, this functionality is often called ETL (extract, transform, and load) in a BI solution. We will discuss the use of SSIS in later chapters.

SQL Server Reporting Services (SSRS): Microsoft has made significant enhancements in the most current version of SSRS to make using this tool an attractive part of a BI solution. One important feature is the visual query designer for SSAS cubes, which facilitates rapid report creation by reducing the need to write manual queries against cube data. The new Report Builder, version 3.0, includes the Report Part Gallery, and enhancements to the Report Manager and Report Viewer components. We will discuss reporting clients, including SSRS, in Chapter 10.

Excel: Many companies already own the Microsoft Office suite, so using Excel as a BI client is often attractive for its low cost and relatively low training curve. We will compare various client solutions in Chapter 10.

PowerPivot: PowerPivot is an exciting new addition to Microsoft's BI tool belt. This new Excel 2010 component enables you to perform data analysis and data mining from your workstation. PowerPivot allows you to work with large amounts of data, perform your analysis, and share your results with others in your company. We will introduce PowerPivot to you in Chapter 12.

SharePoint: This is an optional component to your BI solution. SharePoint is Microsoft's workgroup collaboration and web publishing server. Most easily used in conjunction with SSRS, using the freely available SSRS web parts, SharePoint can expand the reach of your BI solution. As mentioned previously, we will detail options using different BI clients in Chapter 10.

■ **Note** A web part is a pluggable UI showing some bit of content. It is installed globally in the SharePoint portal server web site and can be added to a portal page by any user with appropriate permissions.

Visio: This Microsoft modeling tool for BI projects is optional as well; you can use any tool that you are comfortable using. Sections in Chapter 2 that concern modeling for OLAP include sample Visio diagrams. As with other products in the Office suite, Microsoft has increased the BI integration capabilities with Visio.

Key feature differences between SSAS editions (Standard, Enterprise, and so on) are discussed throughout the entire book. These differences are significant and affect many aspects of your BI solution design, such as the number of servers, number and type of software licenses, and server configuration. You may be thinking at this point, "Wow, that's a big list. Am I required to buy (or upgrade to) all of those

Microsoft products to implement a BI solution for my company?” The answer is no; the only server that is *required* is SSAS. Many companies also provide tools that can be used in a Microsoft BI solution. Although we will occasionally refer to some third-party products in this book, we will primarily focus on using Microsoft’s products and tools to build a BI solution.

BI Languages

An additional consideration is that you will use at least three languages when working with SSAS. The first, which is the primary query language for cubes, is *not* the same language used to work with SQL Server data (T-SQL). The query language for SSAS is called *Multidimensional Expressions* (MDX). SSAS also includes the capability to build data mining structures. To query the data in these structures, you’ll use yet another language—*Data Mining Extensions* (DMX). Finally, Microsoft introduces an administrative scripting language in SSAS—*XML for Analysis* (XMLA). Here’s a brief description of each language.

MDX: This is the language used to query OLAP cubes. Although this language is officially an open standard, and some vendors outside of Microsoft have chosen to adopt parts of it into their BI products, the reality is that very few developers are proficient in MDX. A mitigating factor is that the need for you to manually write MDX in a BI solution can be relatively small—not nearly as much T-SQL as you would manually write for a typical OLTP database. Retaining developers who have at least a basic knowledge of MDX is an important consideration in planning a BI project. MDX is introduced in Chapter 13. Figure 1–4 shows a simple example of an MDX query in SQL Server Management Studio (SSMS).

DMX: This is the language used to query data mining structures (which contain data mining models). Although this language is officially an open standard, and some vendors outside of Microsoft have chosen to adopt parts of it into their BI products, the reality is that very few developers are proficient in DMX. A mitigating factor is that the need for DMX in a BI solution is relatively small (again, not nearly as much T-SQL as you would manually write for a typical OLTP database). Also, Microsoft’s data mining interface is heavily wizard driven, more than creating cubes (which is saying something!). Retaining developers who have at least a basic knowledge of DMX is an important consideration in planning a BI project that will include a large amount of data mining. DMX is introduced briefly in Chapter 14.

XMLA: This is the language used to perform administrative tasks in SSAS. Here are some examples of XMLA tasks: viewing metadata, copying, backing up databases, and so on. Although this language is officially an open standard, and some vendors outside of Microsoft have chosen to adopt parts of it into their BI products, the reality is that very few developers are proficient in XMLA. A mitigating factor is that Microsoft has made generating XMLA scripts simple. In SSMS, when connected to SSAS, you can right-click any SSAS object to generate XMLA scripts using the GUI interface. XMLA is introduced in Chapter 15.

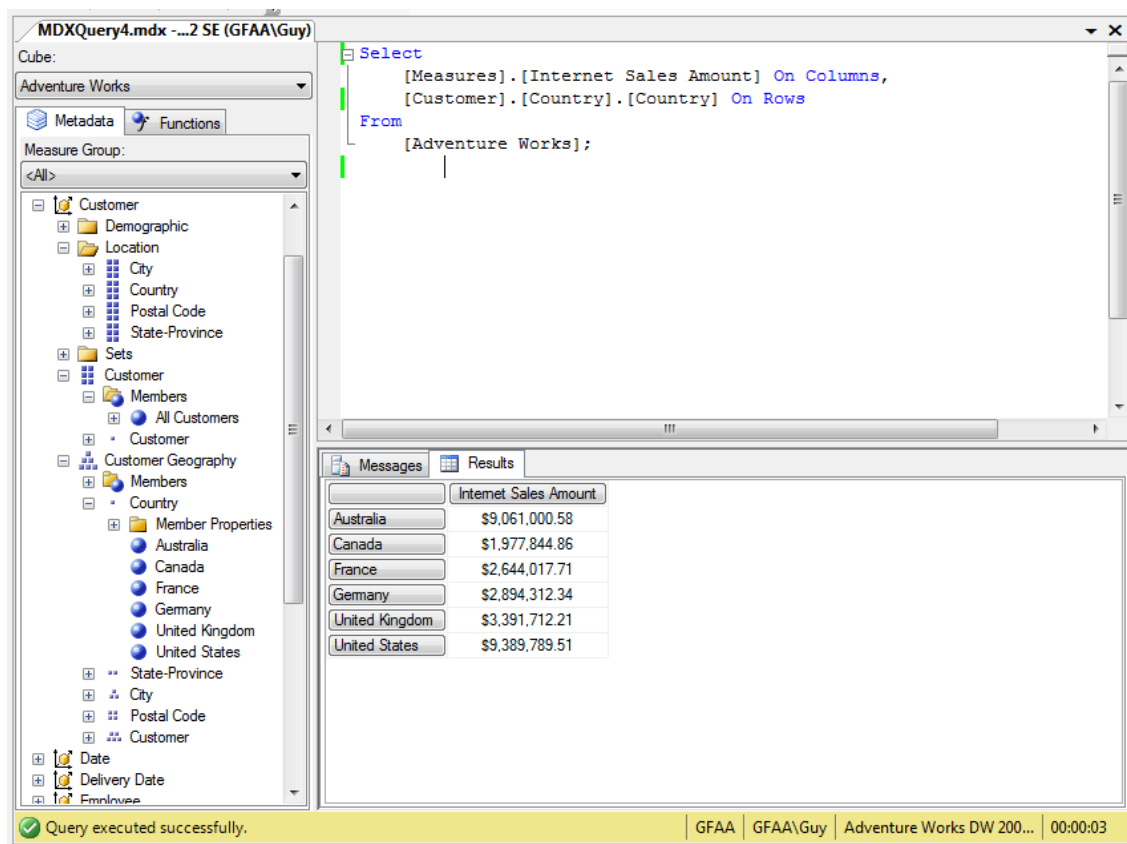


Figure 1–4. The MDX query language is used to retrieve data from SSAS cubes. Although MDX has a SQL-like structure, MDX is far more difficult to master because of the complexity of the SSAS source data structures—cubes.

Because we’ve covered so many acronyms in this section, and we’ll be referring to these products by their acronym going forward in this book, a quick list is provided in Figure 1–5.

| Acronym | Name |
|---------|--|
| BI | Business Intelligence |
| BIDS | Business Intelligence Development Studio |
| BSM | Business Scorecards Manager Server |
| DMX | Data Mining Extensions |
| DSS | Decision Support System |
| HOLAP | Hybrid OLAP |
| MDX | MultiDimensional Expressions |
| MOLAP | Multidimensional OLAP |
| MOSS | Microsoft Office SharePoint Server |
| OLAP | Online Analytical Processing |
| OLTP | Online Transaction Processing |
| PPS | Performance Point Server |
| ROLAP | Relational OLAP |
| SPS | SharePoint Portal Server |
| SSAS | SQL Server Analysis Services |
| SSIS | SQL Server Integration Services |
| SSMS | SQL Server Management Studio |
| SSRS | SQL Server Reporting Services |
| VS | Visual Studio |
| XMLA | XML for Analysis Services |

Figure 1–5. For your convenience, the various BI acronyms used in this book are listed here.

Understanding BI from an End User’s Perspective

You may be wondering where to start at this point. Your starting point depends on the extent of involvement you and your company have had with BI technologies. Usually you will either be completely new to BI, new to SSAS, new to Microsoft’s BI (that is, you are using another vendor’s products to support BI). If BI is new to you and your company, a great place to start is with the end user’s perspective of a BI solution. To do this, you will use the simplest possible client tool for SSAS—an Excel pivot table. This is a great way to familiarize not only yourself but also other members of your team and your executive sponsors about basic BI concepts. Although using Excel may seem like a strange way to showcase a suite of products as powerful as Microsoft’s BI toolset, our experience has shown over and over that this simple approach is quite powerful.

There are two ways to implement the initial setup. Which you choose will depend on the amount of time you have to prepare and the sophistication level of your audience. The first approach is to create a cube using the sample database (AdventureWorksDW 2008 R2) that Microsoft provides with SSAS. Detailed steps for using the first approach are provided later in this chapter. The second approach is to take a very small subset of data from your company and use it for a demonstration or personal study. If you want to use your own data, you’ll probably have to read a bit more of this book to be able to set up a basic cube using your own data.

The rest of this chapter will get you up and running with the included sample. At this point, we are going to focus simply on clicks, that is, “click here to do this.” We are not yet focusing on the reasons behind those clicks. The rest of the chapters will explain in detail just what all this clicking actually does and why you click where you are.

Building the First Sample—Using AdventureWorks

To use the SQL Server AdventureWorksDW 2008 R2 sample database as the basis for building an SSAS cube, you'll need to have at least one machine with SQL Server 2008 R2 and SSAS installed on it. The AdventureWorksDW 2008 R2 sample database is available from CodePlex, at <http://msftdbprodsamples.codeplex.com>. On this page, navigate to the SQL Server 2008 R2 product sample databases. The product sample database page contains a download link for the AdventureWorks examples, as well as download links to view prerequisites and step-by-step installation instructions. While installing, make note of the edition of SQL Server 2008 R2 that you are using (you can use the Developer, Standard, or Enterprise editions), because you'll need to know the particular edition when you install the sample cube files.

To create the sample cube, you will use the sample AdventureWorks Analysis Services project. The sample consists of a set of physical files that contains metadata that SSAS uses to structure the sample Adventure Works cube. As mentioned earlier, you'll work with these sample files in BIDS. The sample is available in the Standard and the Enterprise editions; select the sample file from the directory that matches the edition that you have installed. There are feature differences between the two editions, which you will learn about in detail as you work through the available features in this book.

■ **Note** The Developer edition has an identical feature set to the Enterprise edition (for the purposes of your development, demonstration, or personal review). If you have installed the Developer edition, select the sample from the Enterprise edition folder.

Deploying the Standard Edition Version of the Sample Cube

To deploy the Standard edition of the sample cube, follow these steps:

1. Open the SQL Server Business Intelligence Development Studio (BIDS) from the Start menu.
2. From the BIDS menu, click File ► Open Project/Solution.

Browse to C:\Program Files\Microsoft SQL Server\100\Tools\Samples\AdventureWorks 2008R2 Analysis Services Project\standard, select the file Adventure Works.sln, and click Open (see Figure 1-6).

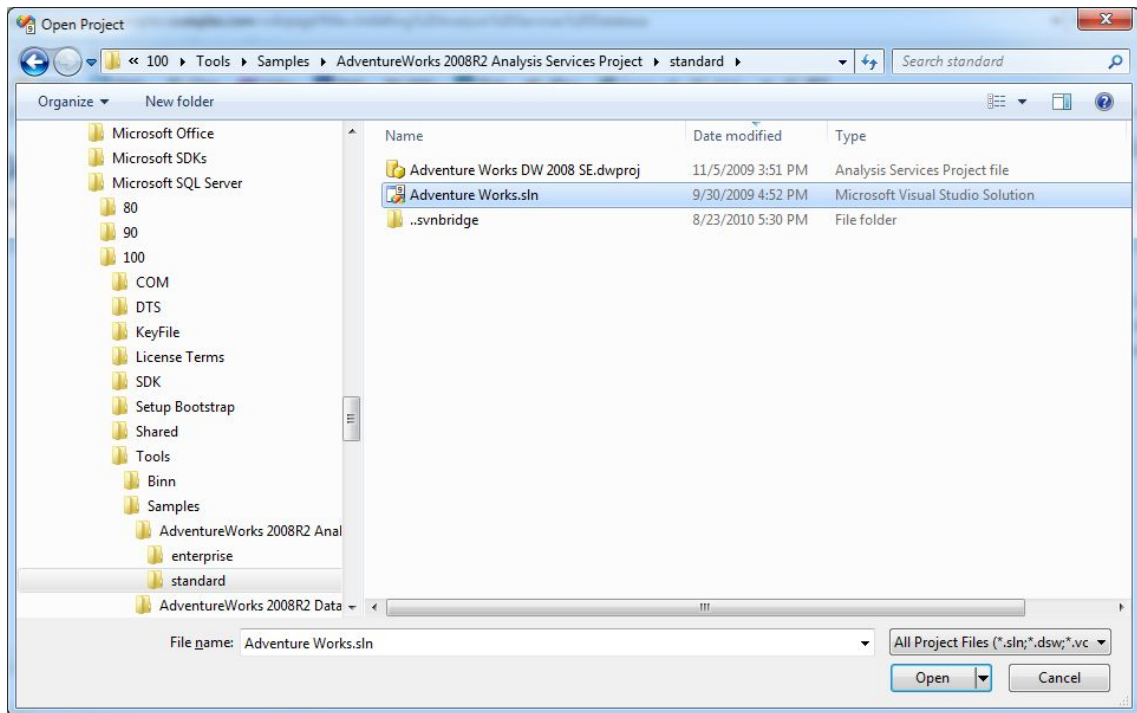


Figure 1–6. To install the SSAS sample cube, select the folder with the edition name that matches the edition of SSAS that you have installed, and then double-click *AdventureWorks.sln* to open the solution in BIDS.

3. Set the connection string to the server name where you deployed AdventureWorksDW by right-clicking the Adventure Works.ds data source in Solution Explorer, and choosing Open. Click the Edit button on the General tab in the Data Source Designer dialog box to change the connection string, shown in Figure 1–7.

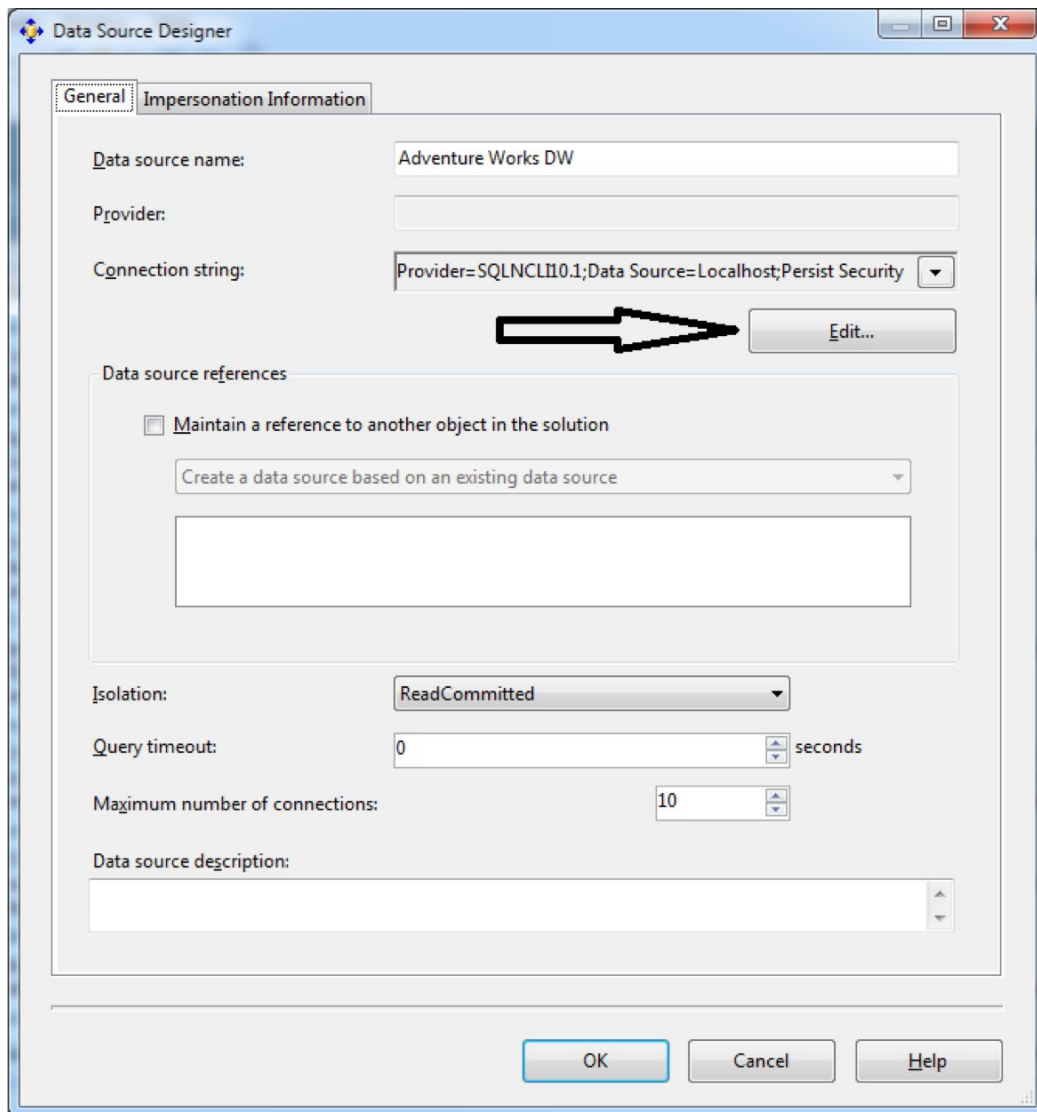


Figure 1–7. When deploying the sample, be sure to verify that the connection string information is correct for your particular installation.

Note If you are using the Enterprise edition, you can follow these steps as well. Simply select the files from the sample Enterprise folder from the path listed next.

4. Be sure to test the connection as well. You do this by clicking the Test Connection button on the bottom of the Connection Manager dialog box, as shown in Figure 1–8.

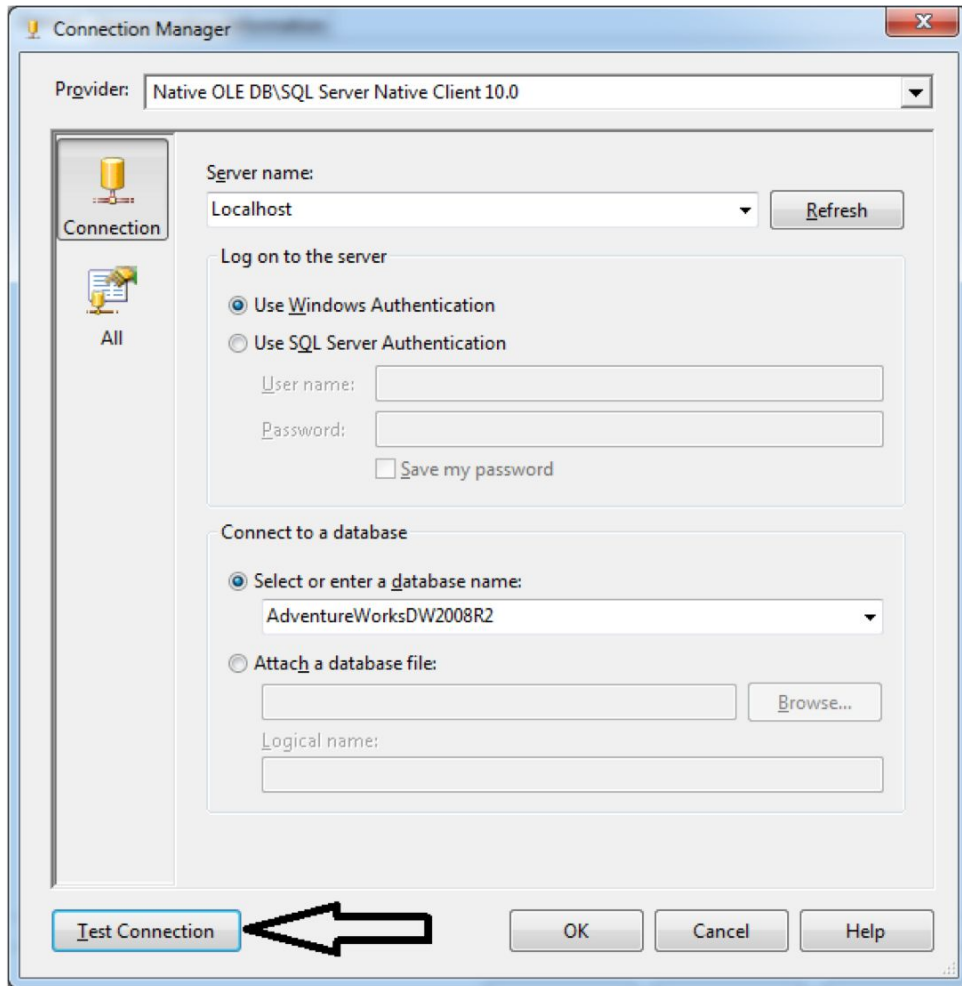


Figure 1–8. You'll want to test the connection to the sample database, AdventureWorksDW2008R2, as you work through setting up the sample SSAS database.

- Right-click the name of the project (Adventure Works DW 2008 SE) in Solution Explorer, and click Properties from the context menu. You must verify the name of the Analysis Services instance that you intend to deploy the sample project to. The default is localhost. If you are using localhost, you do not need to change this setting. You can also use a named server instance, as shown in Figure 1–9. In that case, in the project's Properties Pages dialog box, click Deployment, and set the target sever name to the computer name and instance name separated by a backslash character where you have deployed SSAS (see Figure 1–9). Click OK to accept your entry.

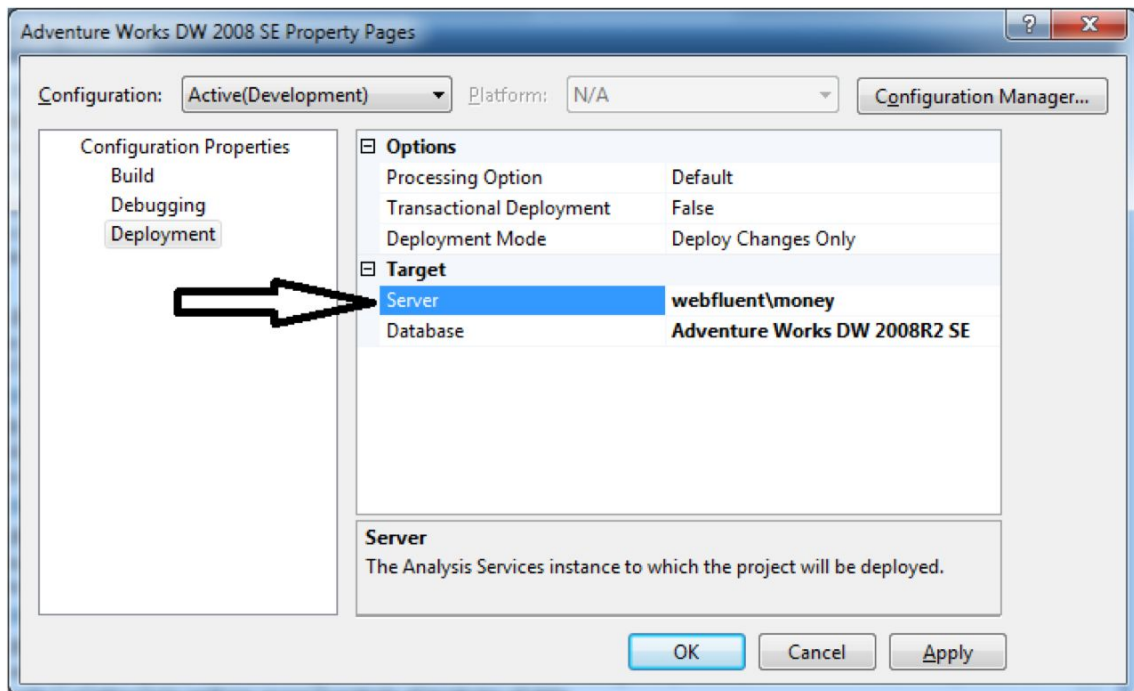


Figure 1–9. Before deploying the sample SSAS project, right-click the solution name in BIDS, and click Properties. In the properties sheet, verify the SSAS instance name.

- From Solution Explorer, right-click the Adventure Works DW 2008 SE project name, and click Deploy. This will process the cube metadata locally and deploy those files to the Analysis Services instance you configured in the previous step.

After clicking Deploy, wait for the “deployment succeeded” message to appear at the bottom right of the BIDS window. This can take five minutes or more depending on the resources available to complete the processing. If the deployment fails (which will be indicated with a large red X in the interface, read the messages in the Process Database dialog box to help you to determine the cause or causes of the failure. The most common error is incorrectly configured connection strings.

Now, you are ready to take a look at the sample cube using the built-in browser in BIDS. This browser looks much like a pivot table so that you, as a cube developer, can review your work prior to allowing end users to connect to the cube using client BI tools. Most client tools contain some type of

pivot table component, so the included browser in BIDS is a useful tool for you. To view the sample cube using the built-in cube browser in BIDS, perform the following steps:

1. In Solution Explorer, expand the Cubes folder, and double-click the Adventure Works cube to open the BIDS cube designer work area (see Figure 1–10).

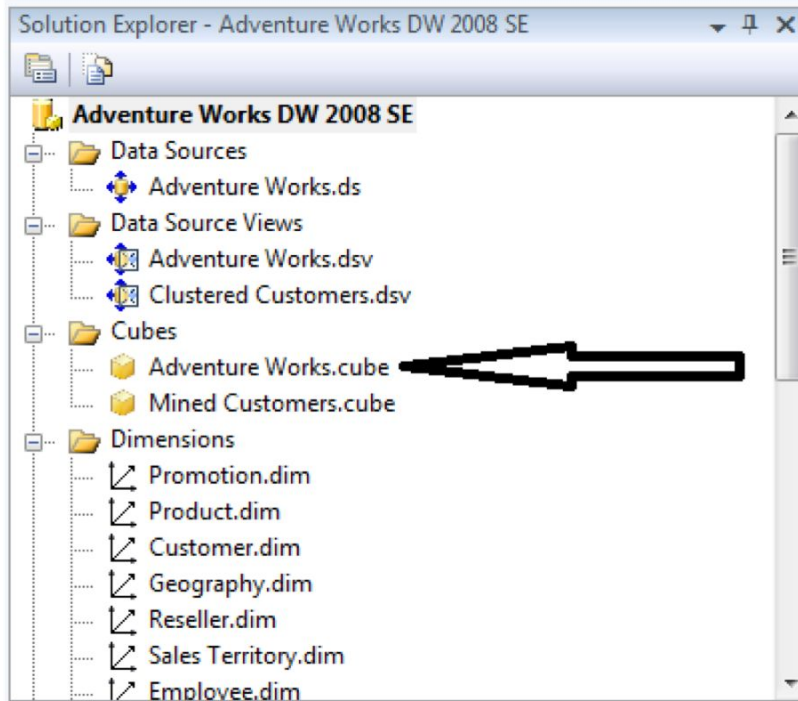


Figure 1–10. To view the sample cube in BIDS, double-click the cube name in Solution Explorer.

2. In the cube designer work area (which appears in the center section) of BIDS, on the AdventureWorks main tab, click the Browser tab as shown in Figure 1–11.

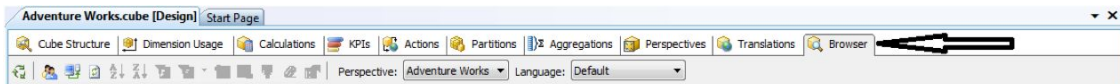


Figure 1–11. The cube designer interface has ten tabs. To browse a cube, you click on the Browser tab. The cube must have been successfully deployed to the server to browse it.

3. Now, you can drag and drop items from the cube (dimensions and facts) onto the viewing area. This is very similar to using a pivot table client to view a cube. The functionality is similar, by design, to BI client tools such as Excel pivot tables. However, the Browser tab, like all of BIDS, is designed for cube designers and *not* for end users.

■ **Note** You may be wondering what the dimensions and facts (or measures) are that you see onscreen? We will review these concepts in more detail in Chapter 2. However, as an introduction, you can think of *facts* as important business values (for example daily sales amount or daily sales quantity), and *dimensions* as attributes (or detailed information) related to the facts (for example, which customers made which purchases, which employees made which sales, and so on).

Spend some time in the BIDS browser interface exploring; drag and drop different items onto the display surface and around the display surface. Also, try right-clicking the design surface to find many interesting built-in options to display the information differently.

You can use Figure 1–12 as a starting point. The Order Count measure is displayed in the data area, the Calendar Year hierarchy from the Date dimension is displayed on the columns axis, the Country hierarchy from the Geography dimension is displayed on rows, the Employee Department attribute from the Employees dimension is displayed as a filter, and the Product Model Categories hierarchy from the Product dimension is set to filter the browser results to include only measure values where the Product Model Category is equal to Bikes.

■ **Tip** To remove any measures or dimensions from the browse area, click the item you want to remove, and drag it back over the tree listing of available objects.

Figure 1–12 is a view of the sample Adventure Works cube. Note that you can place dimension members and hierarchies on the rows, columns, or filter axis and that you can view measures in the area labeled Drop Total or Detail Fields Here.

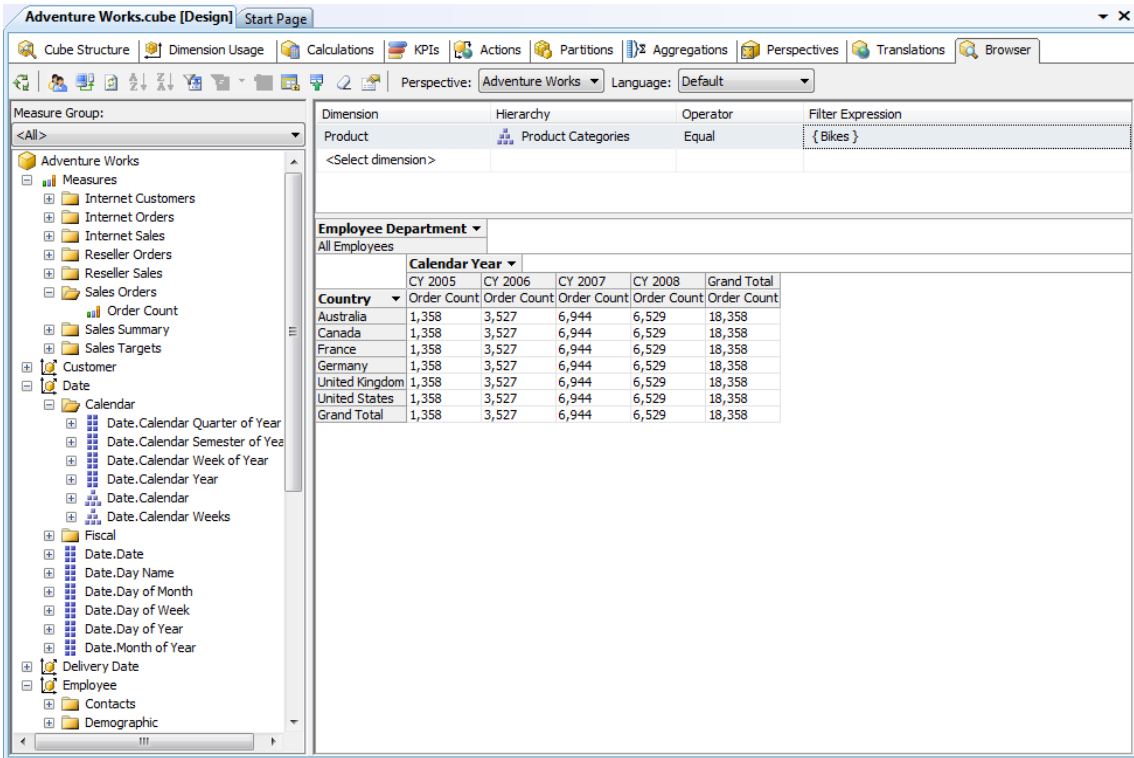


Figure 1–12. The BIDS cube browser uses a pivot table interface to allow you to view the cube that you have built (or, in this case, simply deployed) using the BIDS cube designer.

Note If you are wondering whether you can view sample data mining models in BIDS, the answer is yes. The AdventureWorks samples include data mining structures. Each structure contains one or more data mining models. Each mining model has one or more viewers available in BIDS. Data mining is a deep topic, so we'll spend all of Chapter 14 discussing the mining model types and BIDS interfaces. We discuss Excel support of SSAS mining structures in Chapter 11.

How to Connect to the Sample Cube Using Excel

Now that you've set up and deployed the sample cubes, you will probably want to experience an end user's perspective. An easy way to do this is with a pivot table in Excel:

1. Open Excel.
2. Choose Data ► From Other Sources ► From Analysis Services.

3. In the Data Connection Wizard, enter your SSAS Server name, and click Next, as shown in Figure 1–13.

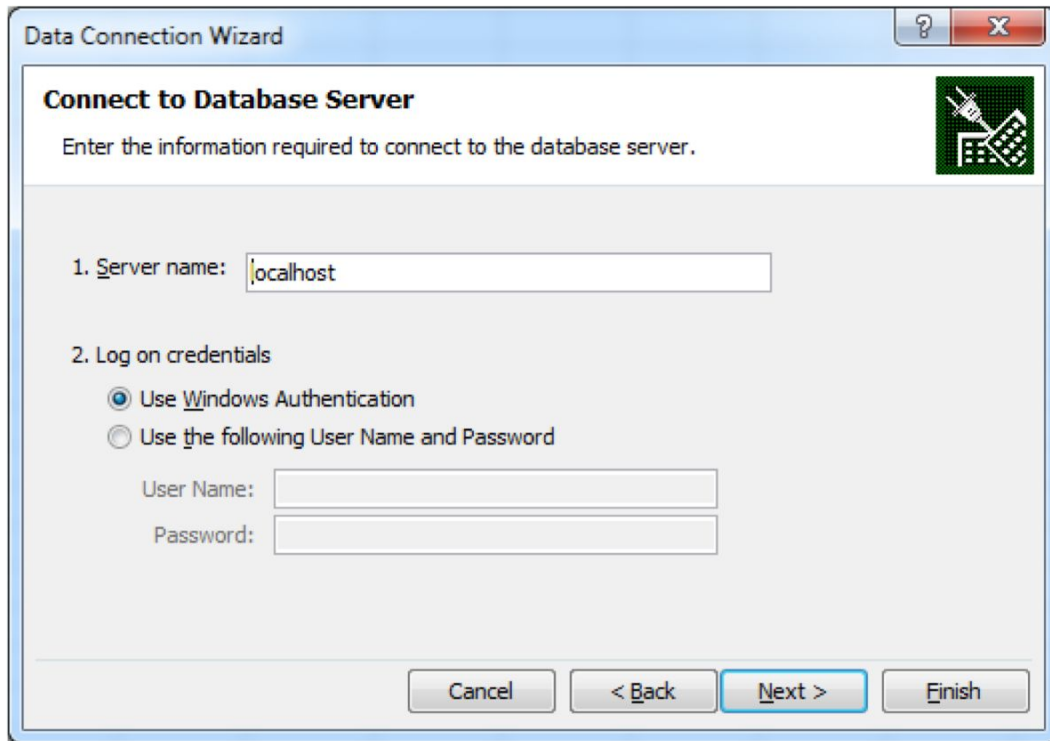


Figure 1–13. Connecting to an SSAS cube in Excel using the Data Connection Wizard

4. Next, select the Adventure Works DW 2008R2 SE database, choose the Adventure Works cube, and click Next, as shown in Figure 1–14.

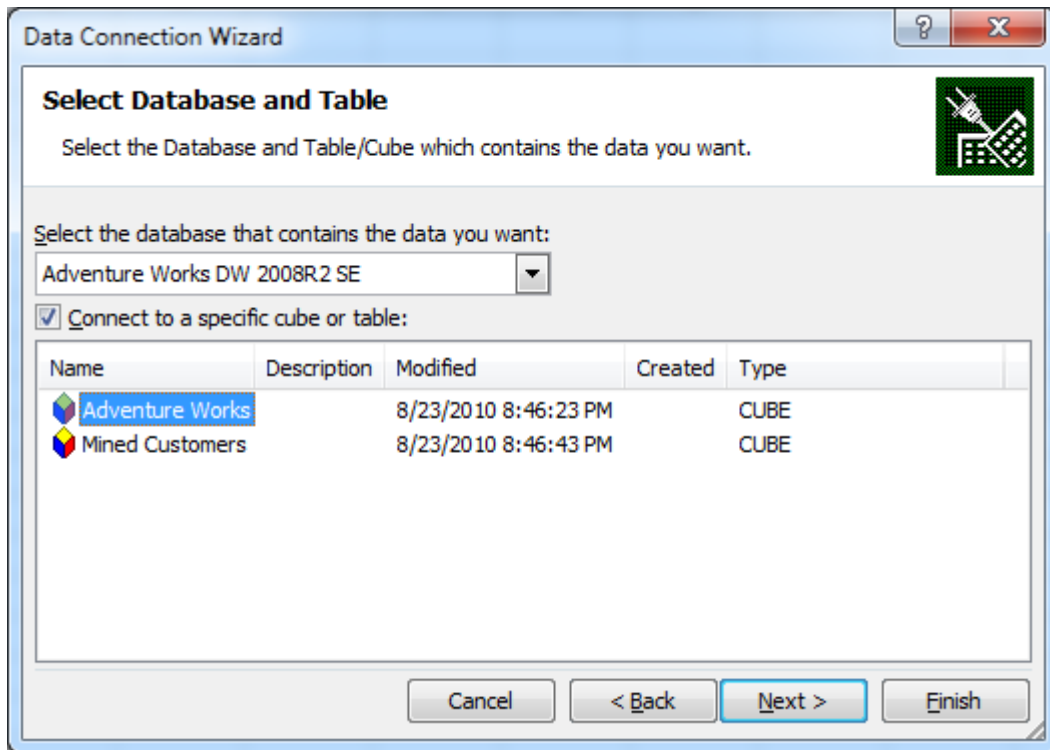


Figure 1–14. Choosing a database and cube using the Data Connection Wizard

5. Review the final wizard dialog, and click Finish.
6. In the Import Data dialog, select PivotTable Report as your data view, and place your data into the existing worksheet with the formula =Sheet1!\$A\$1, and click OK.
7. From the PivotTable Field List, drag the items that you want to show in your pivot table to the Drag Fields area. Figure 1–15 shows an example using Calendar Year as the Column Label, Country as the Row Label, and Customer Count as Values.

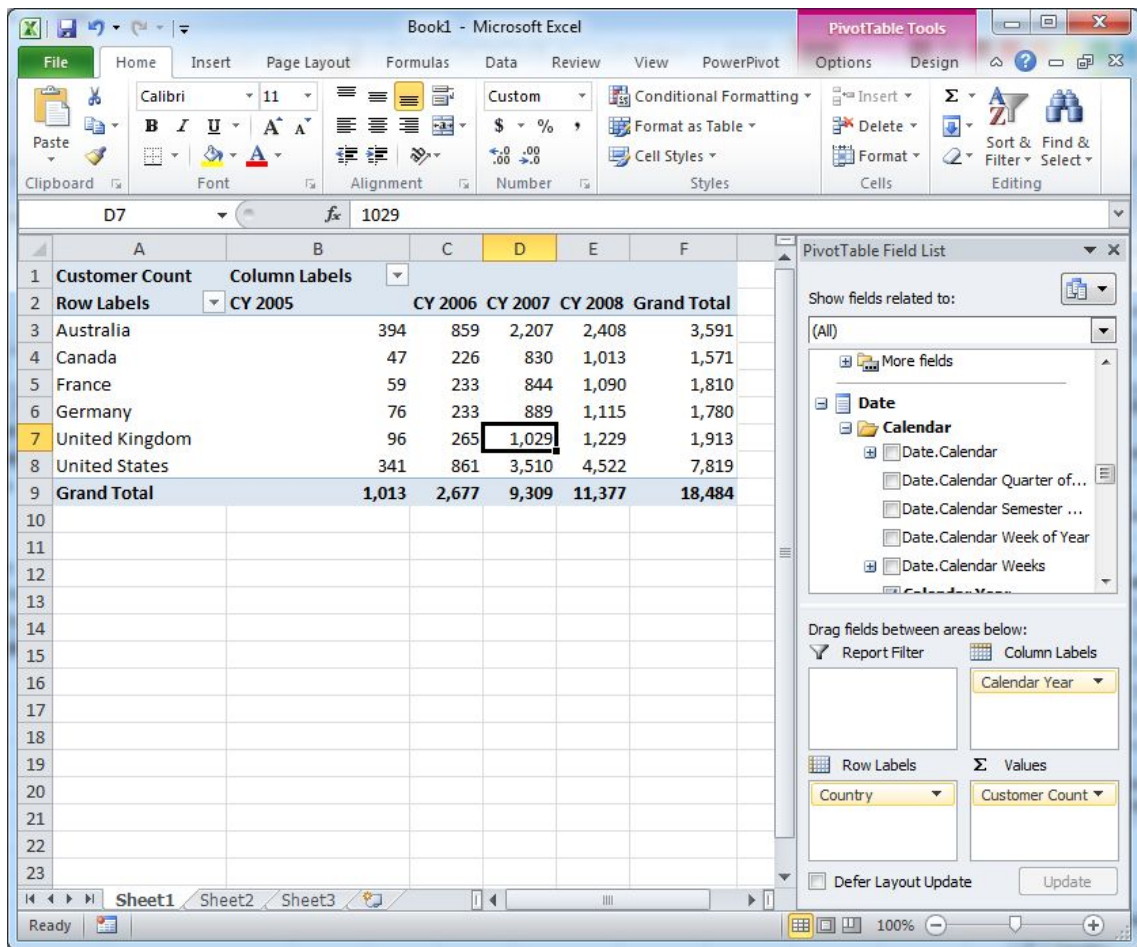


Figure 1-15. An example pivot table using Calendar Year, Country, and Customer Count

Understanding BI Through the Sample

Now that your pivot table is set up, what exactly are you trying to understand by working with it? How is a pivot table that gets its data from a SSAS cube different from any other Excel pivot table? Here is a list of some of the most important BI (or OLAP) concepts:

- BI is *comprehensive* and flexible. A single, correctly designed cube can actually contain all of an organization's data, and importantly, this cube will present that data to end users consistently. To better understand this concept, you should try working with the AdventureWorks sample cube as displayed using the Excel pivot table to see that multiple types of measures (both Internet and Retail Sales) have been combined into one structure. Most, but not all, dimensions apply to both groups of measures. For example, there is no relationship between the Employee dimension and any of the measures in the Internet Sales group because there are no employees involved in these types of sales. Cube modeling is now flexible enough to allow you to reflect business reality in a single cube. In previous versions of SSAS and in other vendor's products, you would've been forced to make compromises such as creating multiple cubes or being limited by structural requirements. This lack of flexibility in the past often translated into limitation and complexity in the client tools as well.
- BI is *accessible* (intuitive for all end users to view and manipulate). To better understand this aspect of BI, try demonstrating the pivot table based on the SSAS sample cube to others in your organization. They will usually quickly understand and be impressed (some will even get excited!) as they begin to see the potential reach for BI solutions in your company. Pivot table interfaces reflect the way many users think about data; they answer the question, "what are the measures (or numbers) and what attributes (or factors) created these numbers?"

Some users may request a simpler interface than a pivot table (that is, a type of canned report). Microsoft provides client tools, such as SSRS, which facilitate that type of implementation. It is important for you to balance this type of request, which entails manual report writing by you, versus the benefits available to end users who can use pivot tables. In my experience, most BI solutions include a pivot table training component for those end users who haven't worked much with pivot tables before.

- BI is *fast to query*. After the initial setup, queries can easily run 1,000% faster in an OLAP database than in an OLTP database. Your sample won't necessarily demonstrate the speed of query in and of itself. However, it is helpful to understand that the SSAS server is highly optimized to provide a far superior query experience because the SSAS engine itself is actually designed to quickly fetch or calculate aggregated values. We will dive into the details of this topic in Chapter 6.
- BI is *simple to query*. End users simply drag items into and around the pivot area; developers write very little query code manually. It is important to understand that SSAS clients (like Excel) automatically generate MDX queries when users drag and drop dimensions and measures onto the design surfaces. This is a tremendous advantage as compared to traditional OLTP reporting solutions where T-SQL developers must manually write all of the queries.

- BI provides accurate, near real-time, *summarized* information. This will improve the quality of business decisions. Also with some of the new features available in SSAS, most particularly Proactive Caching, cubes can have latency that is only a number of minutes or even seconds. Configuring real-time cubes will also be discussed in Chapter 6. Also, by drilling down, users who need to see the detail (that is, the numbers behind the numbers) can do so. *Drilldown* is, of course, implemented in pivot tables via the simple “+” interface that is available for all (summed) aggregations in the AdventureWorks sample cube.
- BI *improves ROI* by allowing more end users to make more efficient use of enterprise information so many companies have all the information they need. The problem is that the information is not accessible in formats that are useful for the people in the company to use as a basis for decision making in a timely way.

It’s really just that simple; OLAP (or BI) solutions simply give businesses a significant competitive advantage by making more information available to more end users so that those users can make better decisions in a more timely way. What’s so exciting about BI is that Microsoft has made it possible for many companies who couldn’t previously afford to implement any type of BI solution to be able to play in this space by including all of the core BI tools and technologies needed to implement cubes in the box with SQL Server. As previously stated, it is important to understand which features require the Enterprise edition of SQL Server or Analysis Services. We will review feature difference by edition in detail throughout this book.

In addition to broadening BI’s reach by including some BI features in both the Standard and Enterprise editions of SQL Server, Microsoft is also providing some much needed competition at the enterprise level by including some extremely powerful BI features in the Enterprise editions of SQL Server and SSAS. We’ll talk more about that at the end of this chapter.

Understanding the Business Problems That BI Addresses

As you learn more about SSAS capabilities, you can begin to match some of the strengths of the BI toolset available in SQL Server (and companion BI Microsoft products) to current challenges you and your company may be facing when working with your enterprise data. We call these challenges *pain points* and list some OLAP (or BI) solutions to commonly seen challenges in Table 1–1.

Table 1–1. List of Business Pain Points and OLAP Solutions

| Pain Point | OLAP Solutions |
|---|---|
| Slow-to-execute queries | Use cubes that are optimized for read-only queries to return query results faster than OLTP systems because of the efficiency of the SSAS engine to aggregate data. |
| General system slowdowns | Greatly reduce locking overhead from OLTP source systems. OLAP systems use optimized, multi-dimensional storage that do not use locks, except during processing. |
| Manual query writing | Allow end users to click to query (click and drag on a pivot table), which eliminates the wait time associated with traditional T-SQL, where end users must usually request reports, which results in developers manually writing queries against OLTP systems. |
| Disparate data sources | Combine data into central repositories or cubes using ETL packages that can be automated to run on a regular basis. |
| Invalid/inconsistent report data | Invalid reports are often based on data that has been cleaned and validated (prior to cube load) using the ETL toolset available in SSIS. Cubes provide a consistent and unified view of enterprise data all across the enterprise. |
| Data is not available to all users | Data is designed to be accessed by all business users. |
| Too much data | Use data mining (along with the other tools, that is, cubes, available in SSAS) to find patterns in large amounts of data automatically. SSAS now contains nine different data mining algorithms to help you group, correlate, and predict data values. |

Reasons to Switch to Microsoft's BI Tools

In addition to providing a great suite of tools for companies that are just getting started with BI, Microsoft's latest release of BI tools also targets companies that are using other vendors' BI products by providing a raft of enterprise features in its data warehousing products. Many of these features are available only in the Enterprise edition of the various BI products, that is, Analysis Services, Integration Services, and Reporting Services.

Here's a list of BI-specific features that require the Enterprise edition of SQL Server:

- Advanced business analytics
- Proactive caching
- Advanced data management
- Write back

- Advanced data mining tuning
- Advanced SSIS transforms
- Text mining support

Also, Microsoft has built its BI tools so that they will integrate with other vendors' products. It is quite common, for example, to use SSAS to create cubes from Oracle or DB2 data sources. Another example is to use SSRS with a mainframe or an Informix source data. Microsoft is aggressively adding support for interoperability across the entire suite of BI tools. Another compelling aspect of Microsoft's BI offering is the inclusion of intelligent wizards and GUI tools that allow you to get up and running quickly and easily. The catch, however, is that the use of these tools and wizards is heavily dependent on your understanding and implementation of basic OLAP modeling concepts. We will look at that topic in the next chapter.

Summary

This chapter introduced basic data warehousing terms and concepts, including OLAP, BI, dimensions, and facts. We also reviewed the process and procedures you use to quickly set up a sample SSAS cube using BIDS. You worked with the AdventureWorks sample and connected to it with an Excel pivot table to give you a quick view of an OLAP solution from an end user's perspective. In the next chapter, we'll dig deeper into OLAP concepts and explore basic modeling for cubes.



OLAP Modeling Concepts

You've got executive support and a great BI team assembled. You've diligently set up standards and practices. The development environment is set up (and secured!), and your team is ready to start designing your solution. What is the next step? It all starts with a star—a star schema, that is.

Properly designed OLAP schemas are the foundation of all successful BI projects built using SQL Server Analysis Services (SSAS). With star schemas as a starting point, you or your ETL gurus can begin the data mapping process, and you or your report writers can begin to create report prototypes. This chapter will explain design models for OLAP schemas—stars, snowflakes, and more—all of which are the basis for OLAP cubes. Modeling for data mining is *not* covered here because it is discussed in Chapter 14, “Introduction to Data Mining.”

This chapter covers the following topics:

- Modeling OLAP source schemas: stars, snowflakes, and other types
- Understanding dimensional modeling, including modeling for changing dimensions
- Understanding fact (or measure) and cube modeling
- A quick introduction to other types of modeling, such as KPIs, Actions, and Translations

Modeling OLAP Source Schemas—Stars

Learning about OLAP modeling always starts with a thorough review of the “classic” OLAP source model: the star schema. The next section reviews the concepts behind star schema modeling in detail.

Before we start however, let's take a minute to discuss an even more fundamental idea for your BI project. Is a star schema strictly required? The technical answer to this question is “no” because Microsoft purposely does *not* require you to base OLAP cubes off of only data that is in a star schema format. In other words, you can create cubes based off of OLTP (or normalized relational data).

However, and this is a *big* however, it is critical for you to understand star schema modeling and to attempt to provide SSAS with data that is as close to this format as possible. The reason for this is that, although Microsoft *has* included flexibility in SSAS, that flexibility is really designed for those of you who intend to create cubes manually. Particularly if you are new to BI (using OLAP cubes), you'll probably want to build your first project by taking advantage of the included wizards and tools in BIDS. These time savers are designed to work using traditional star schema source data. When you become more experienced, you will probably find yourself enjoying the flexibility to go “outside the star” and then will build some parts of your BI solution manually.

Understanding the Star Schema

A *star schema* consists of at least one fact table and a number of dimension tables. These tables are relational database tables, often stored in SQL Server. The star schema source tables are not required to be stored in SQL Server. In fact, many OLAP solutions use other relational database management system (RDBMS) platforms, including Oracle, DB2, and others, to hold source star data.

A star schema fact table consists of at least two types of columns: keys and measures. *Keys* are foreign key (FK) values that relate rows in the fact table to rows in the dimension tables. *Facts* (which may also be called *measures*) are numeric values—usually, but not always, additive—that express the business metrics. An example of this is a sales amount (fact) for a particular product, sold on a particular day, by a particular employee (keys).

Fact tables can also contain columns that are neither keys nor facts. These columns are the basis for a special type of dimension called a *degenerate dimension*. For example, in the fact table in Figure 2–1, the SalesOrderNumber column provides information about each row, but it is neither a key nor a fact. Often, degenerate dimensions are used as a track-back, or pointer, to data in the source system.

Figure 2–1 also shows a typical fact table structure: the first columns are all named xxxKey and of datatype int. These columns are the FK values. They provide the relationship to the dimension tables and are said to “give context or meaning to” the facts. In Figure 2–1, the columns that will be translated into measures in the cube start with the OrderQuantity column. Note the datatypes for the measure columns. You may be surprised to see the use of the SQL Server money datatype. If a fact column represents a monetary value, then it is preferred to use the money datatype in the fact table because some Multidimensional Expressions (MDX) functions are dependent on this datatype.

■ **Note** Facts or measures, what’s the difference? Technically, *facts* are individual values stored in rows in the fact table, and *measures* are those values as stored and displayed in an OLAP cube. The terms are commonly used interchangeably in OLAP literature.

Another important consideration when modeling your fact tables is to keep the tables as narrow as possible. The reason for this is that a star schema fact table generally contains a much larger number of rows than any one-dimensional table. So fact tables represent your most significant storage space concern in an OLAP solution. It is especially important for you to justify every column added to any fact table in your star schemas against your project’s business requirements.

| GFAA.AdventureW...actInternetSales | | SQLQuery1.sql - ...r (GFAA) | |
|------------------------------------|----------------------|-----------------------------|--------------------------|
| | Column Name | Data Type | Allow Nulls |
| ▶ | ProductKey | int | <input type="checkbox"/> |
| | OrderDateKey | int | <input type="checkbox"/> |
| | DueDateKey | int | <input type="checkbox"/> |
| | ShipDateKey | int | <input type="checkbox"/> |
| | CustomerKey | int | <input type="checkbox"/> |
| | PromotionKey | int | <input type="checkbox"/> |
| | CurrencyKey | int | <input type="checkbox"/> |
| | SalesTerritoryKey | int | <input type="checkbox"/> |
| 🔑 | SalesOrderNumber | nvarchar(20) | <input type="checkbox"/> |
| 🔑 | SalesOrderLineNumber | tinyint | <input type="checkbox"/> |
| | RevisionNumber | tinyint | <input type="checkbox"/> |
| | OrderQuantity | smallint | <input type="checkbox"/> |
| | UnitPrice | money | <input type="checkbox"/> |
| | ExtendedAmount | money | <input type="checkbox"/> |
| | UnitPriceDiscountPct | float | <input type="checkbox"/> |
| | DiscountAmount | float | <input type="checkbox"/> |

Figure 2–1. Fact tables consist of keys and facts (or measures) and, sometimes, additional columns, like *SalesOrderNumber*, which provide additional information. The keys are FKs to the dimension table rows. Measures are usually, but not always, additive numeric values.

Understanding a Dimension Table

As mentioned previously, the dimension table rows provide meaning to the rows in the fact table. Each *dimension table* describes a particular business entity or aspect of the fact table entries. Typical dimension tables include time, geography, customers, and products. Dimension tables should consist of three types of columns. The first is a newly generated primary key (PK) for each row in the dimension table. The second is the original PK from the source system, and the third group consists of any number of additional columns that further describe the business entity.

Keys

Dimension tables are not strictly required to contain two types of keys. You could actually create dimension tables using only the original PK; however, this practice is *not* recommended. One reason to generate a new unique dimension key is that it is common to load data into dimensions from disparate data sources (for example, a SQL Server table and an Excel spreadsheet). Without generating new keys, you would have no guarantee of having a unique identifier for each row.

As mentioned previously, you should model for two types of keys (or identifiers) in dimensional source data. The first is the original PK, which is the key from the source system. This is also sometimes called the *business key*. In addition to this key, it is a best practice to generate a new, unique key during

the extract, transform, and load (ETL) process of the dimension table. The new key is called a *surrogate key*.

Even if you are retrieving source data from a single source database initially, it is an important best practice to add this new surrogate key on loading the dimension table. The reason is that business conditions can quickly change—you may find yourself having to modify a production cube to add data from another source for many reasons (business merger, acquisition, competitor data, and so on). You should always use surrogate keys when building dimension tables.

You'll note that the DimCustomer dimension table shown in Figure 2–2 contains both the original identifier, called CustomerAlternateKey, and a new unique identifier called CustomerKey.


| GFAA.Adventure...dbo.DimCustomer | | GFAA.AdventureW... | |
|---|---------------|-------------------------------------|--|
| Column Name | Data Type | Allow Nulls | |
|  CustomerKey | int | <input type="checkbox"/> | |
| GeographyKey | int | <input checked="" type="checkbox"/> | |
| CustomerAlternateKey | nvarchar(15) | <input type="checkbox"/> | |
| Title | nvarchar(8) | <input checked="" type="checkbox"/> | |
| FirstName | nvarchar(50) | <input checked="" type="checkbox"/> | |
| MiddleName | nvarchar(50) | <input checked="" type="checkbox"/> | |
| LastName | nvarchar(50) | <input checked="" type="checkbox"/> | |
| NameStyle | bit | <input checked="" type="checkbox"/> | |
| BirthDate | date | <input checked="" type="checkbox"/> | |
| MaritalStatus | nchar(1) | <input checked="" type="checkbox"/> | |
| Suffix | nvarchar(10) | <input checked="" type="checkbox"/> | |
| Gender | nvarchar(1) | <input checked="" type="checkbox"/> | |
| EmailAddress | nvarchar(50) | <input checked="" type="checkbox"/> | |
| YearlyIncome | money | <input checked="" type="checkbox"/> | |
| TotalChildren | tinyint | <input checked="" type="checkbox"/> | |
| NumberChildrenAtHome | tinyint | <input checked="" type="checkbox"/> | |
| EnglishEducation | nvarchar(40) | <input checked="" type="checkbox"/> | |
| SpanishEducation | nvarchar(40) | <input checked="" type="checkbox"/> | |
| FrenchEducation | nvarchar(40) | <input checked="" type="checkbox"/> | |
| EnglishOccupation | nvarchar(100) | <input checked="" type="checkbox"/> | |
| SpanishOccupation | nvarchar(100) | <input checked="" type="checkbox"/> | |
| FrenchOccupation | nvarchar(100) | <input checked="" type="checkbox"/> | |
| HouseOwnerFlag | nchar(1) | <input checked="" type="checkbox"/> | |
| NumberCarsOwned | tinyint | <input checked="" type="checkbox"/> | |

Figure 2–2. Dimension tables contain denormalized source data. Dimensions give context and meaning to the facts in the fact table. It is typical for dimension tables to contain columns for many types of attributes.

Attributes

You may be surprised to see the large number of columns in the `DimCustomer` table. The denormalized style of modeling is completely opposite of that which you were probably taught when learning database modeling for OLTP systems. That's actually the point—OLAP modeling is quite different from modeling for OLTP!

Your SSAS cube can contain tens, hundreds, or even thousands of attributes to describe the business entities. The attributes are built from the source columns in the dimension source table or tables. Although most limits to the quantity of dimensional attributes that you can associate to a particular business entity have been removed in SSAS, you do want to base the inclusion of columns in your dimension tables on business needs. In our real-world experience, this value is usually between 10 and 50 attributes per dimension.

Unlike erring on the conservative side (as we recommend you do when modeling the fact table)—that is, if in doubt, leave it out—when you are modeling dimensions, we recommend the opposite approach. If there is a possibility that a particular attribute will be of interest to a set of your end users, then add it to your dimension. It is trivial to include, rename, or even exclude attributes when building your cube. Unless you anticipate that your dimension will be huge, having for example, more than a million members (which sometimes can be found in a dimension for customers, for example), then being “inclusive” in dimensional modeling is preferred.

There are additional options to OLAP modeling (that is, using table types other than fact tables and star dimension tables), which we will discuss later in this chapter, but the basic concept is simply a fact table and some related dimension tables. Figure 2–3 shows a conceptual star schema; note that we've modeled the dimensions keys in the preferred way in this diagram, that is, using original and new (or surrogate) keys.

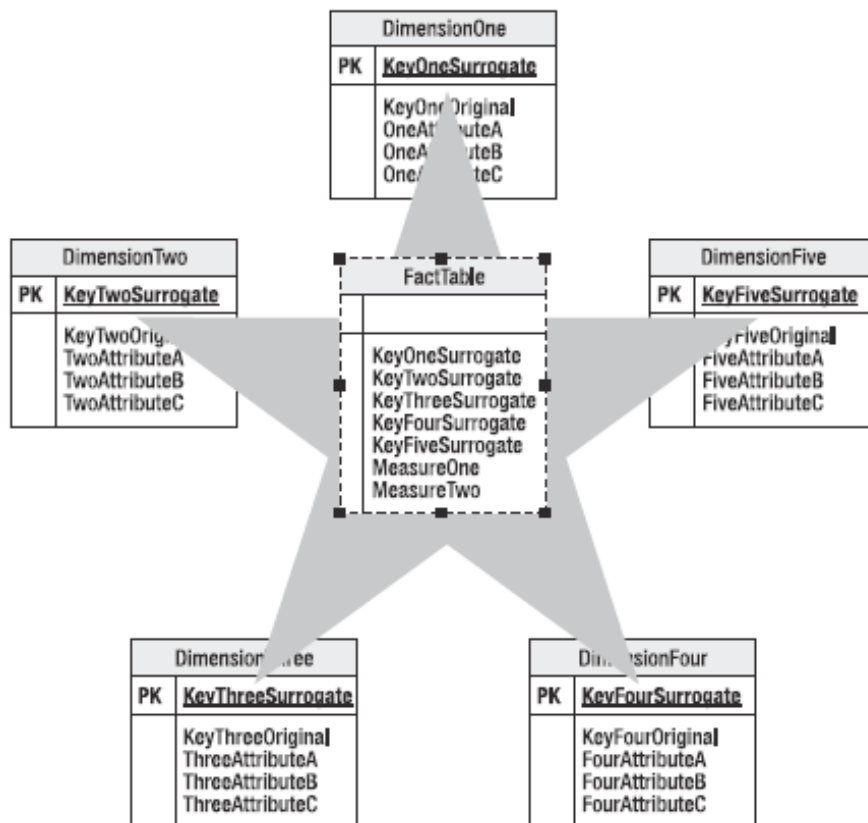


Figure 2-3. A star schema consists of at least one fact table and many dimension tables. The dimension table rows are related to the fact table rows by surrogate keys.

Why Create Star Schemas?

As discussed earlier, the simplest answer is that star schemas work best in the BIDS development environment for SSAS. Although it is possible to create a cube from OLTP (or normalized) source data, the results will not be optimal without a large amount of manual work on your part, and we do not recommend this practice. Also, flawed data in source systems is commonly discovered during the life cycle of a BI project. Usually at least some source data needs to be part of a data cleansing and validation process. This is performed during the ETL phase of your project. Again, we've seen many a BI project "go astray" because source data was assumed to be "perfect" and found, upon investigation, to be far from that.

Beginning with the 2005 release of SSAS, Microsoft improved the flexibility of source structures for cubes. This means you start with a series of star schemas and then make adjustments to your model to allow for business situations that fall outside of a strict star schema model. One example of this is the ability to base a single cube on multiple fact tables. Figure 2-4 shows an example of using two fact tables in an OLAP schema. This type of modeling is usually done because some, but not all, dimensions relate to some, but not all, facts. You'll see an example of this shortly.

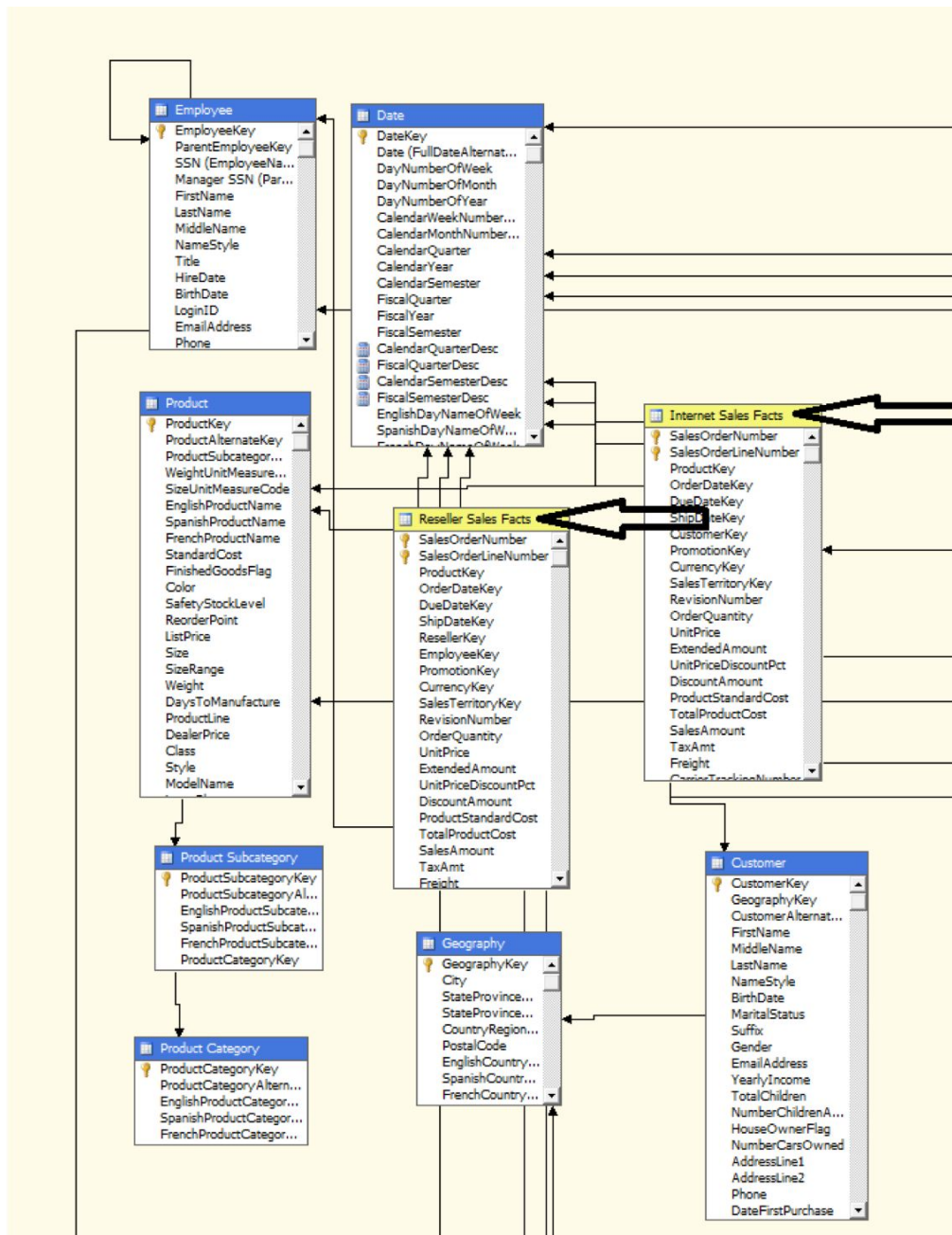


Figure 2–4. SSAS cubes can use more than one fact table as part of the source data, which results in greater flexibility in cube building.

As will be discussed later in this chapter, there is flexibility in SSAS that allows you to model and implement in your OLAP cube which reflect many common business scenarios and which are not part of star schemas. Just one example of this is the need to allow null values to be loaded into a dimension or fact table and to define a translation of those nulls into a value that was understandable by end users, for example, “unknown.” This flexibility is not so broad however, as to eliminate the need to do standard OLAP modeling entirely.

The SSAS Dimension Usage tab in the cube designer in BIDS allows you to define the grain of each relationship between the rows in the various fact tables to the rows in the dimension tables. This improved flexibility now results in most BI solutions being based on a single, large (or even huge) cube (or view of enterprise data). This type of modeling reflects the business need to have a single, unified version of relevant business data. This cube presents the data in whatever level of detail is meaningful for the particular end user; that is, it can be summarized, or detailed, or any combination of both. This ability to create one view of the (business) truth is one of the most compelling features of SSAS.

To drill into the Dimension Usage tab, look at Figure 2–5. Here the Employee dimension has no relationship with the Internet Sales facts (because no employees are involved in Internet sales) but does have a relationship with the Reseller Sales facts (because employees are involved in reseller sales). Also, the Customer dimension has no relationship with the Reseller Sales facts because customers are not resellers, but the Customer dimension does have a relationship with the Internet Sales facts (because customers do make Internet purchases). Dimensions common to both fact tables are products and time (due date, order date, and ship date). Note that the time dimension has three aliases. Multiple aliases for a single dimension are called *role-playing dimensions*. We’ll discuss this type of dimension in more detail in Chapter 5, “Advanced OLAP Modeling with SSAS.”

| Measure Groups | | |
|----------------------|------------------------|------------------------|
| Dimensions | Internet Sales | Reseller Sales |
| Date | Date | Date |
| Date (Ship Date) | Date | Date |
| Date (Delivery Date) | Date | Date |
| Customer | Customer | |
| Reseller | | Reseller |
| Geography | | Reseller |
| Employee | | Employee |
| Promotion | Promotion | Promotion |
| Product | Product | Product |
| Sales Territory | Sales Territory Region | Sales Territory Region |

Figure 2–5. The Dimension Usage grid in SSAS allows you to associate more than one fact table with a single cube, to set the level of granularity for each dimension, and to choose not to associate specific dimensions with specific fact tables.

When creating a cube, the SSAS New Cube wizard attempts to detect the relationships between dimension and fact table rows and populates the Dimension Usage grid with its best guesses by examining the source column names. You should review (and update if needed) the results to exactly match your particular business scenarios.

Effectively Creating Star Schema Models Using Grain Statements

So, if the star schema is all-important, what's the best way for you to quickly and accurately create this model? In our experience, if you *begin with the end* in mind, you'll arrive at the best result in the quickest fashion. *The end* is a series of *grain statements*. So what exactly does this mean? To determine (or validate) grain statements, you can ask the following questions:

- What are the *key metrics* for your business? Some examples for a company that sells products include sales amount, sales quantity, gross profit, net profit, expenses, and so on.
- By what *factors* do you evaluate those key metrics? For example, do you evaluate sales amount by customer, by employee, by store, by date, by "what"?
- By what *level of granularity* do you evaluate each factor? For example, do you evaluate sales amount by day or by hour? Do you evaluate customers by store or by region?

Effective OLAP modelers use the grain statements gathered during the requirements phase of the project. It is critical that both subject matter experts (SMEs) and business decision makers validate each of the grain statements prior to beginning the modeling phase. We use a sign-off procedure to ensure that appropriate validation of grain statements has taken place.

Here are some examples of simple grain statements:

- We want to see sales amount and sales quantity by day, by product, by employee, and by store location.
- We want to see average score and quantity of courses taken, by course, by day, by student, by manager, by curriculum, and by curriculum type.
- We want to see a count of offenders by location, by offense type, by month, and by arresting officer.

■ **Caution** You might be tempted to skip entirely or to move too quickly through the requirements gathering/modeling phases of your BI project to get to the "real work" of actually building the cube in BIDS. We've seen this mistake repeated many, many times in the real world. The results are, at best, longer-than-needed development cycles, particularly in cases where people try to build cubes directly off of relational source (or slightly modified) data, or, at worst, projects have to be restarted from scratch. Don't make this costly mistake!

As you can see by my examples, BI solutions can be used by a broad variety of organizations. In our experience, although the "show me sales by day" model is the most typical, it's not the only situation in which BI can prove useful. Some other interesting projects we've worked on included using OLAP cubes to improve decision support for the following business scenarios:

- Improve detection of foster care families not meeting all state requirements (SSAS data mining was also used in this scenario)
- Provide a flexible, fast query system to look up university course credits that are transferable to other universities
- Improve food costs and labor costs for a restaurant by viewing and acting on both trends and exception conditions
- Track the use and effectiveness of a set of online training programs by improving the timeliness and flexibility of available reports

When considering why and where you might implement SSAS in your enterprise, it is important to think broadly across the organization; that is, you should consider which groups would benefit from an aggregated view of their (and possibly other groups') data. It is Microsoft's position, and we agree, that any organization with stored data can benefit from using BI implemented on SSAS OLAP cubes.

Tools for Creating Your OLAP Model

Our Microsoft modeling tool of choice for designing star schemas is Visio Enterprise Edition. It's easy to use, readily available, and can be used to quickly generate the T-SQL data definition language (DDL) source statements so that your design for star schemas can be rapidly materialized on your development server.

You'll usually start your design with dimension tables because much of the dimension data will be common to multiple grain statements. In the example we've provided in Figure 2–6, you can see that the relatively few tables are highly denormalized (meaning they contain many columns with redundant data; for example in *StudentDim*, the *region*, *area*, and *bigArea* columns).

Contrast this type of design with OLTP source systems, and you'll begin to understand the importance of the modeling phase in an OLAP project. In Figure 2–6, each dimension source table, except two (*OfferingDim* and *SurveyDim*), is the basis of a single cube dimension; for example, *StudentDim* is the basis of the *Student* dimension, *InstructorDim* is the basis of the *Instructor* dimension, and so on. These are all examples of star dimensions. *OfferingDim* and *SurveyDim* have a PK/FK relationship between the rows. They are the basis for a snowflake (or multitable sourced) dimension. You'll learn more about snowflake dimensions later in this chapter.

Also notice in Figure 2–6 that each table has two identity (or key) fields: a *NewID* and an *OldID*. This is modeled in the preferred method discussed earlier in this chapter.

Figure 2–7 shows the fact tables for the same project. You can see that there are nearly as many fact tables as dimension tables in this particular model example. This isn't necessarily common in OLAP model design; more commonly, you'll use 1 to 5 fact tables with 5 to 15 dimension tables, or more of both types. This model is used to illustrate reasons for using multiple fact tables; for example, some Session types have facts by day, whereas other Session types have facts by hour.

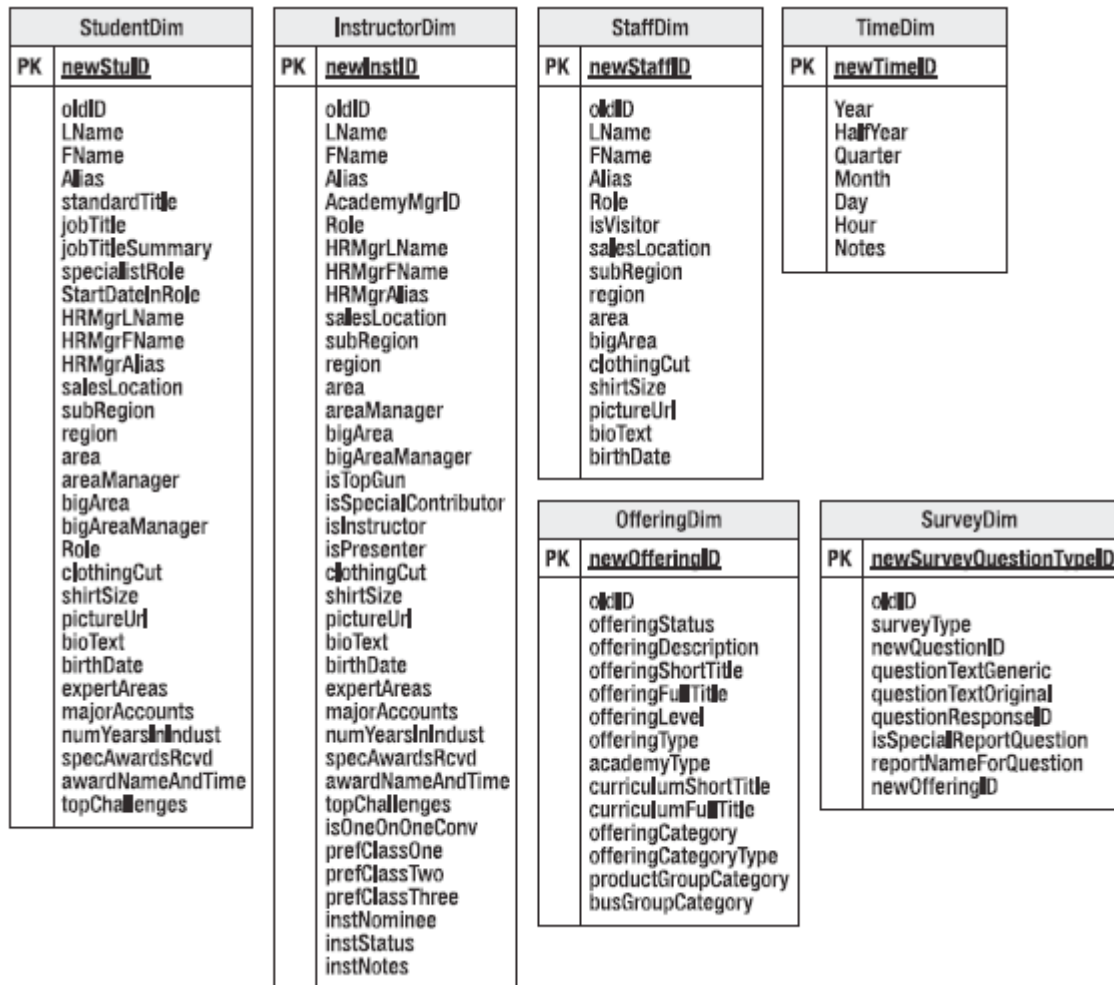


Figure 2-6. This model shows mostly star-type (or single table) dimensions. There is one snowflake-type (or multitable) dimension: SurveyDim and OfferingDim.

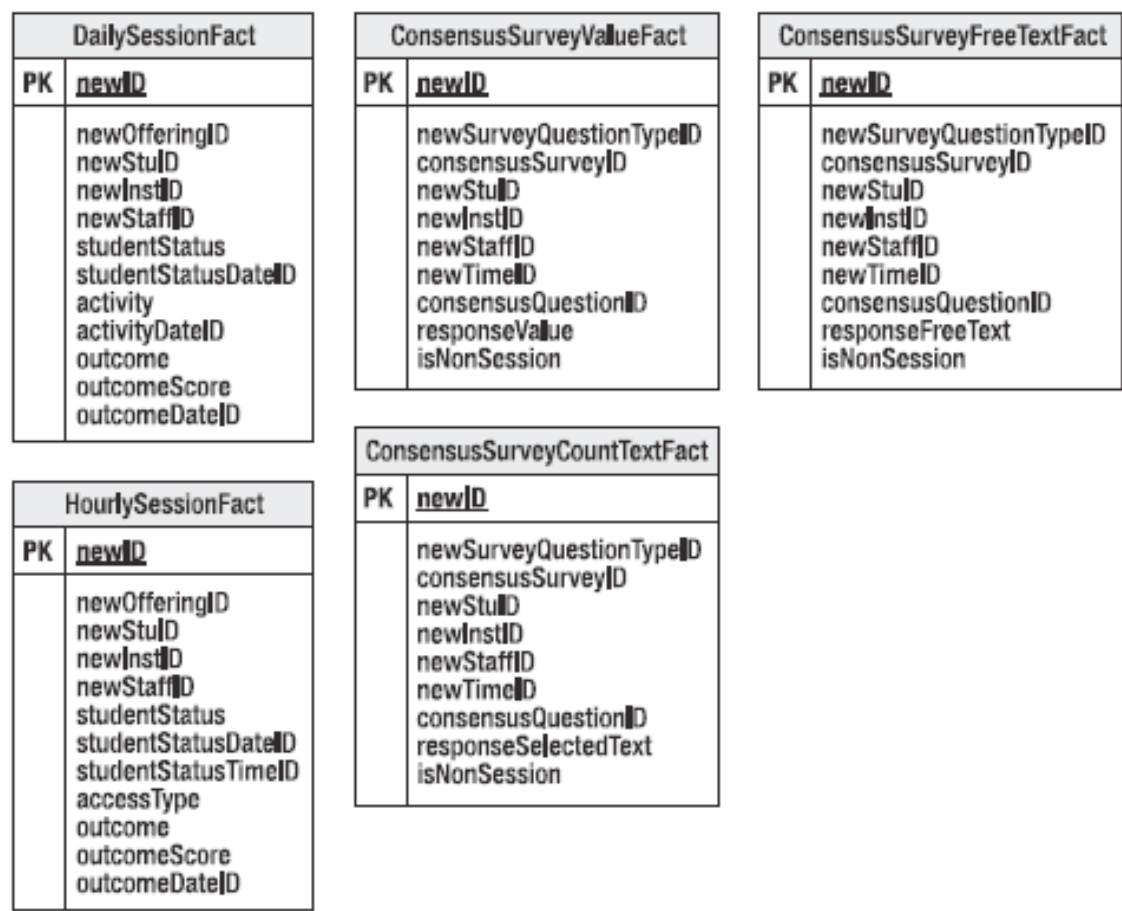


Figure 2-7. This model shows five fact tables that will be used in a single cube.

Tip Because your project is now in the development phase, any and all documents, including your Visio (.vsd) models, must be under source control if multiple people will be working on the OLAP models.

As mentioned previously, although we’ve used Visio for modeling in all of our projects, if you are comfortable with a different database modeling tool, such as ERWIN or ER/Studio, by all means, use it. The primary requirements for your modeling tool is that it generates a visual representation along with DDL code, so anything you are comfortable using for OLTP can be used for OLAP modeling as well.

As in relational modeling, OLAP modeling is an *iterative* process. When you start, you’ll simply create the skeleton tables for your star schema by providing table names, keys, and a couple of essential

column names (such as first name, last name for customer). As you continue to work on your design, you will refine the model by adding detail.

Because it is so critical, look back at the conceptual diagram of an OLAP star schema, shown earlier in Figure 2-3, one more time. Remember that this is the structure that you are trying to emulate. The closer you can get your models to true stars, the more quickly and smoothly the rest of the entire BI project will run.

Also remember the importance of using the customer's business terminology when naming objects in your model. When you name your tables and columns per the captured taxonomy, the your model will be understood, validated, and translated into cubes more quickly and easily by everyone working on your project.

Modeling Source Schemas—Snowflakes and Other Variations

As mentioned previously, SSAS has increased the flexibility of source schema usage to more easily accommodate the most common business requirements that aren't easily modeled using star schemas. This section discusses some of those new or improved options.

Understanding the Snowflake Schema

A *snowflake* is a type of source schema used for dimensional modeling. Simply put, it means basing a dimension on more than one source relational table. The most common case is to use two source tables. However, if more than two tables are used as the basis of a snowflake dimension, there must be a key relationship between each of the tables containing the dimension information.

Note in the example in Figure 2-8 that the Customer dimension has a GeographyKey attribute in it. This allows for the snowflake relationship between the the Geography and the Customer dimensions to be detected by the New Cube wizard in BIDS. The Dimension Usage section of SSAS usually reflects the snowflake relationship, which you have modeled when you initially create the cube using the New Cube wizard (as long as the key columns have the same names across all related tables). If necessary, you can manually adjust any relationships after the cube has been created using tools provided in BIDS.

As shown previously in this chapter, Figure 2-9 again shows the Dimension Usage grid. This time, we are going to drill down a bit deeper into using it. To adjust, or verify any relationship, click the Build button (the small grey square with the three dots on it) on the dimension name at the intersection of the dimension and fact tables. We'll start by looking at a "regular" or star dimension by clicking the Build button at the intersection of the Product dimension and Internet Sales facts.

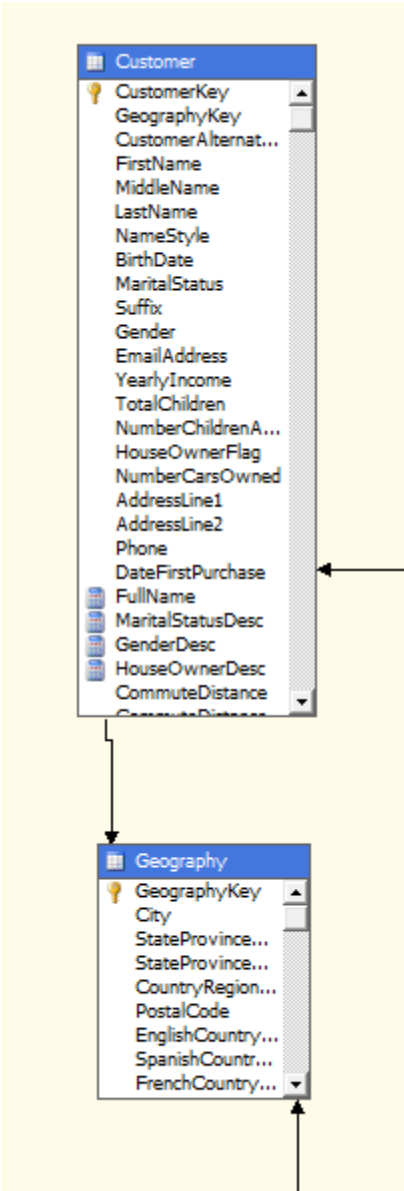


Figure 2–8. The customer dimension uses two tables as its sources: Customer and Geography. The rows in the tables are related by the GeographyKey field.

| Dimensions | Measure Groups | | |
|----------------------|------------------------|------------------------|------------------------|
| | Internet Sales | Internet Orders | Internet Customers |
| Date | Date | Date | Date |
| Date (Ship Date) | Date | Date | Date |
| Date (Delivery Date) | Date | Date | Date |
| Customer | Customer | Customer | Customer |
| Reseller | | | |
| Geography | | | |
| Employee | | | |
| Promotion | Promotion | Promotion | Promotion |
| Product | Product | Product | Product |
| Sales Territory | Sales Territory Region | Sales Territory Region | Sales Territory Region |
| Source Currency | Source Currency Code | Source Currency Code | Source Currency Code |
| Sales Reason | Sales Reasons | Sales Reasons | Sales Reasons |

Figure 2–9. The Dimension Usage grid is the starting point for defining the nature of the relationships between the dimension and fact tables. This includes defining the level of granularity of each relationship.

Clicking the Build button opens the Define Relationship dialog box in which you can confirm that the relationship that BIDS detected during cube build is correct. If the relationship has been incorrectly defined, you can adjust it here. In Figure 2–10, you can see that a Regular (or star) relationship has been correctly detected in the Select Relationship Type drop-down list—you validate this by verifying that the correct identifying key columns have been detected by BIDS when the cube was initially created. In this example, using the ProductKey from the Product dimension (as PK) and Internet Sales fact tables (as FK) reflects the intent of the OLAP design.

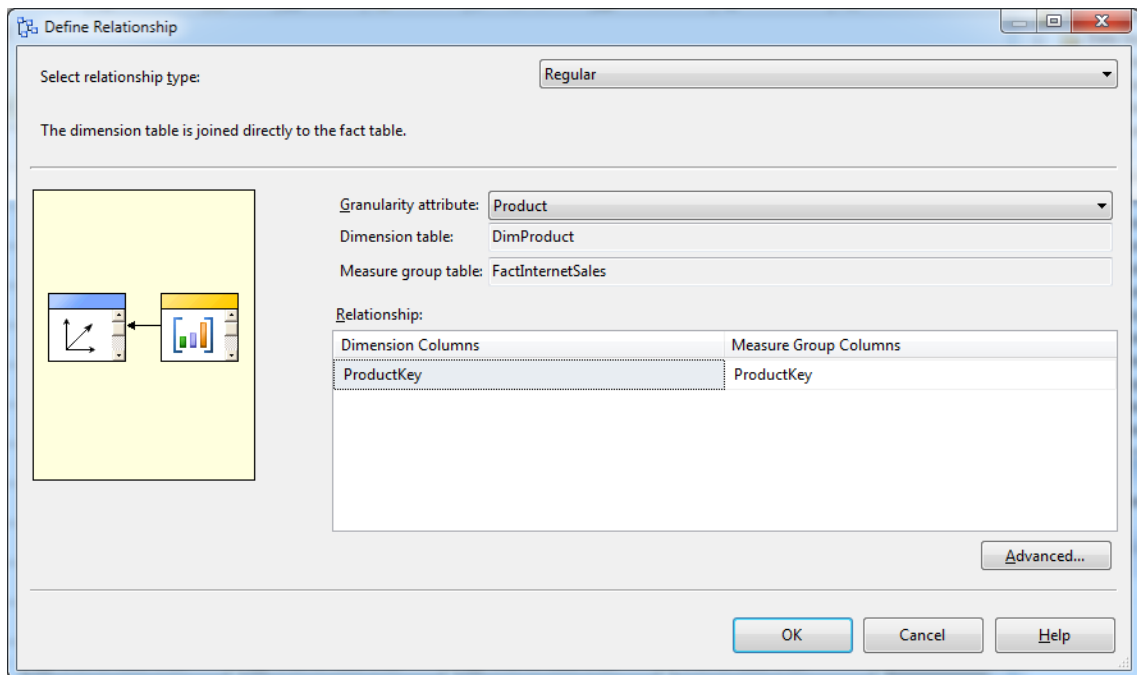


Figure 2–10. The most typical relationship type between the dimension and fact tables is the Regular (or star) type. This means there is a zero or one-to-many relationship between the rows in the fact table and the dimension table, based on the listed key.

For a snowflake (or referenced) dimension, you review or refine the relationship between the related dimension tables in the Define Relationship dialog box as shown in Figure 2–11. Note that the dialog box changes to reflect the modeling needs; that is, you must select the intermediate dimension table and define the relationship between the two dimension tables by selecting the appropriate key columns.

Note You will generally leave the Materialize check box checked (the default setting) for snowflake dimensions. This causes the value of the link between the fact table and the reference dimension for each row to be stored in SSAS and improves dimension query performance.

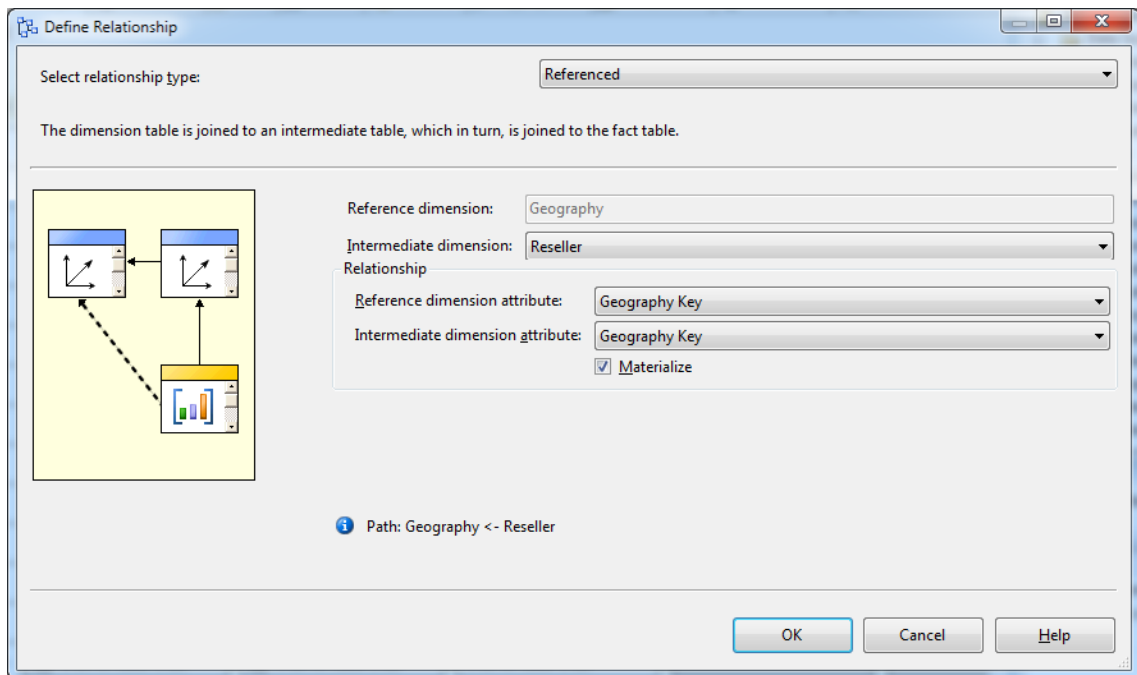


Figure 2-11. Another possible relationship type between the dimension and fact tables is the *Referenced* type. This means that there is a one-to-many relationship between the rows in the fact table and the dimension table and includes an additional table, still based on the listed key. This is also called a *snowflake dimension*.

Knowing When to Use Snowflakes

Because snowflakes add overhead to cube processing time and to query processing time, you should only use them when the business needs justify their use. They add overhead because the data must be joined at the time of process or query, rather than simply retrieved. The most typical business situation that warrants the use of a snowflake dimension design is one that would reduce the size of the dimension table by removing one or more attributes that are not commonly used to a separate dimension table. An example of this would be a customer dimension with an attribute (or some attributes that are used for a small percentage of the customer records).

An example of this might be a table of nonessential customer data, such as a URL of a customer's web site in a business scenario where very few of your customers actually have their own web sites. By creating a separate but related table, you significantly reduce the size of the customer dimension table. Another situation that may warrant the use of a snowflake design is one in which the update behavior of particular dimensional attributes varies; that is a certain set of dimensional attributes should have their values overwritten if updated values become part of the source data, whereas a different set should have new records written for each update (maintaining change history). Although it is possible to combine different types of update behavior depending on the complexity of the dimension, it may be preferred to separate these attributes into different source tables so that the update mechanisms can be simpler.

■ **Tip** In the real world, we've often seen inexperienced OLAP modelers overuse snowflakes. It is important to remember that the primary goal of the star schema is to denormalize the source data for read efficiency. Any normalization, such as a snowflakes dimension, should relate directly to business needs. As this is opposite of OLTP modeling, it's often difficult to fight the urge to normalize. Our experience is that less than 15% of dimensions need to be presented as snowflakes.

Considering Other Possible Variations

With SSAS, there are several new techniques that OLAP modelers can use. These include Many-to-Many dimensions, Data Mining dimensions, and more. These (and other) more advanced modeling techniques are discussed in Chapter 8, "Intermediate SSIS," and Chapter 14, "Introduction to Data Mining."

Choosing Whether to Use Views Against the Relational Data Sources

At this point, you may be thinking that this OLAP modeling seems like a great deal of work, so why not just create views against the OLTP source (or sources) to get the same result? Although you technically could do this, as previously mentioned, our experience has been that seldom are the relational source or sources "clean" enough to directly model against. The most typical situation is that first the OLAP model is created and validated, and then cleaned and validated data is loaded into the newly created OLAP model via ETL processes. Most organizations' data simply isn't prepared to allow for direct OLAP query against OLTP source data.

One area where relational views are sometimes used in OLAP projects is as data sources for ETL. That is, in environments where OLAP models and ETL engineers are not allowed direct access to data sources, it is common for them to access the various data sources via views created by DBAs. Also the time involved to write the queries to be used in the relational views may be substantial.

Understanding Unified Dimensional Modeling

The Unified Dimensional Model (UDM) is one of the key features of Analysis Services. In addition to removing the requirement that each dimension's hierarchies (or rollups) must be defined only at time of creation, Microsoft has simplified dimensional modeling by basing all dimensions on attributes. Simply put, each column from the source dimension table is (by default) an attribute for that dimensional item.

A *hierarchy* in SSAS is a grouping mechanism, and serves two purposes. The first is largely a convenience for end users. This type of hierarchy is called a *browse* (or *navigational*) *hierarchy*. The second type is called a *natural hierarchy* and will be discussed in greater detail in Chapter 3, "Introducing OLAP Modeling with SSAS." Natural hierarchies can change how aggregations (or precalculated intersections of facts) are created. Also, all dimensional information is public, or sharable across all cubes.

Using the UDM

Understanding the “how” of modeling dimensional data, including understanding dimensions, levels, members, and hierarchies, it’s best to start with some definitions (see Table 2–1) and a couple of examples. Figure 2–12 shows the Dimension Structure work area in BIDS.

Table 2–1. List of OLAP Terms

| Term | Definition | Example |
|-----------|--|-------------------------|
| Dimension | Entity and all attributes related to that entity | Customers |
| Hierarchy | Grouping of attribute values for an entity | Customers by Geography |
| Member | Instance of entity, including attributes | ID=50, Name = Langit... |
| Level | Name of rollup position in hierarchy | State Level |
| Key | Primary identifier, two types: surrogate (or new) and original | NewID = 1, OldID = 101 |

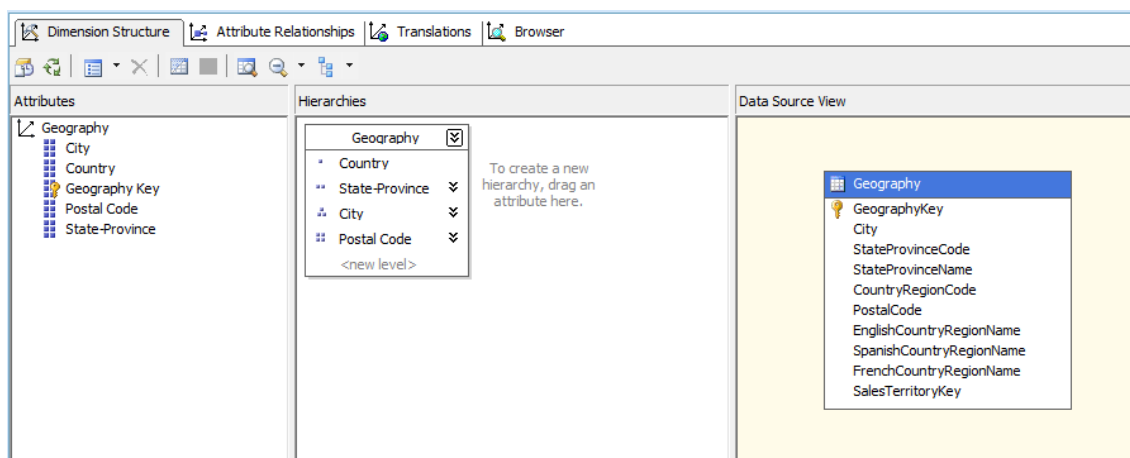


Figure 2–12. The Dimension Structure work area in BIDS allows you to view and edit dimension information.

The most important initial OLAP modeling consideration is to make every attempt to denormalize all source data related to a particular entity. As previously stated, the preferred source design for OLAP is the star schema, which means that each dimension’s source data is put into a single table. Typically, these tables are very wide, that is, having a large number of columns, and not especially deep, that is not having a large number of rows.

An example of this might be a product dimension. Your company may sell only a couple hundred different types of products; however, you may retain many, many attributes about each product. Some examples could include package size, package color, introduction date, and so on. There can be

exceptions to the general “wide, but not deep” modeling rule. The most common is for the customer’s dimension. If it is a business requirement to capture all customers for all time, and if your organization services a very large customer base, then it could be the case that your customer dimension source table could have millions of rows. Analysis Services only loads dimension members being viewed by your client tool into memory. This allows you to be inclusive in the design of dimensions, that is, more (attributes) is usually better.

After you’ve created the appropriate source dimension table or tables and populated them with data, SSAS will retrieve information out of these tables during cube and dimension processing. SSAS then uses a `SELECT DISTINCT` statement to retrieve members from each column. If you use the cube wizard to build your cube, Analysis Services will attempt to locate natural hierarchies in the data. A *dimensional hierarchy* is data that has a one-to-many relationship between data in different columns. A typical example of this is a customer’s table with attributes relating to the customer’s address. City, state, and country are detected during cube creation by the wizard and associated into a hierarchy with three levels. If SSAS detects natural hierarchies during the running of the New Cube wizard, it will name each level in the hierarchy using the column names from the dimension source tables. You can easily update these names during subsequent cube development.

When end users browse cube data, they can look at the facts or measures by any attribute value in any dimension by default. This could result in hundreds, thousands, or even millions of data members at each level in a dimension. Without hierarchies, this information could be overwhelming in volume and not especially meaningful when end users try to view the data. These hierarchies are used to aggregate the view of information so that it is more meaningful and useful for end users of the cube. Figure 2–13 shows a common hierarchy, customers by geography. Another way to understand hierarchies is to think of them as summaries. For this particular example, you can look at the information in your cube at the level of the particular location or summarized to the level of all locations in a particular city, all locations in a particular province, or all locations in a particular country.

In addition to the hierarchies that are detected during cube build; you can manually create hierarchies in the cube very quickly and easily. The manual building of both navigations and natural dimensional hierarchies is covered in Chapter 3, “Introducing OLAP Modeling with SSAS.”

There are a couple of other new features in SSAS dimensions that you should consider when you are in the modeling phase of your BI project. These features include the ability to set the default member for each attribute in a dimension, the ability to convert nulls to a value (usually unknown or 0), and the ability to allow duplicate names to be displayed. You should model for these features based on business requirements, so you should capture the business requirements for defaults, unknowns, and duplicates for each dimension and all of its attributes during the modeling phase of your project.

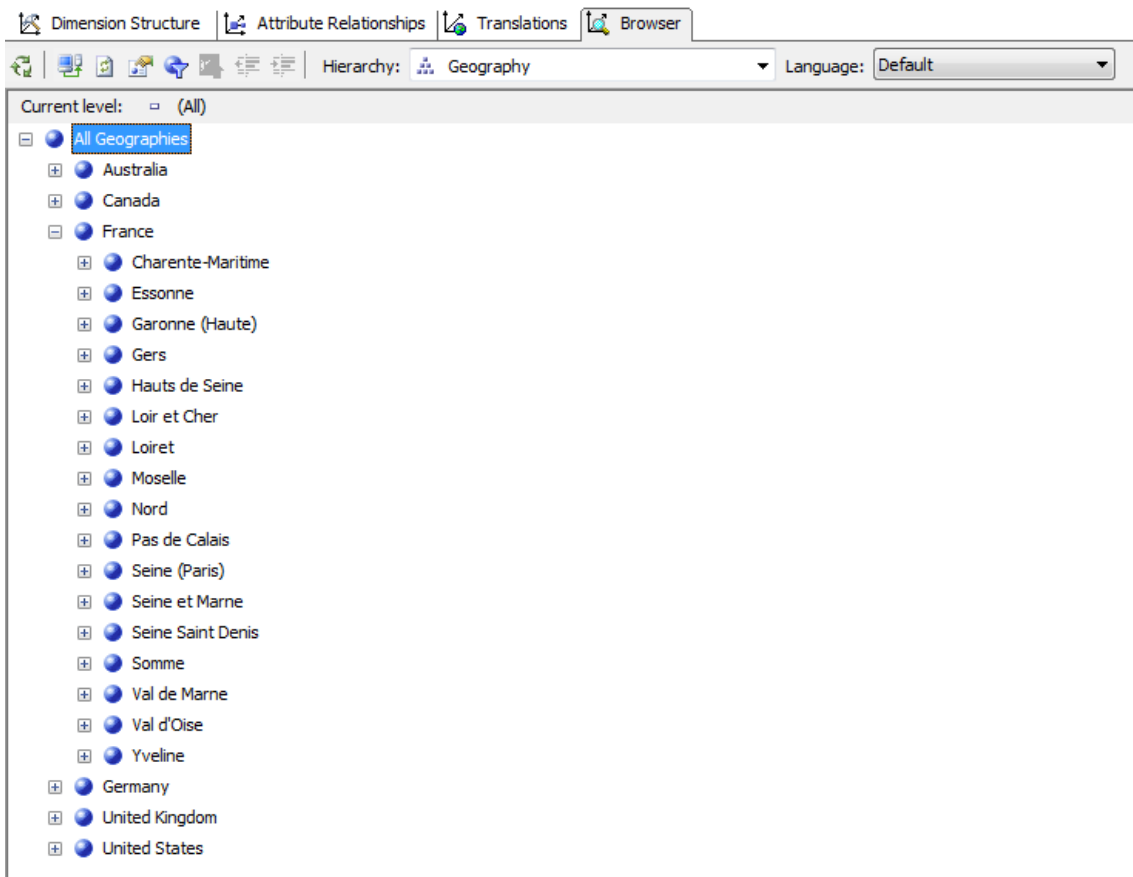


Figure 2–13. The Dimension Designer in SSAS includes a Browser tab that lets you review the structure of the dimension by hierarchy (if hierarchies exist) prior to building your cube. If there are no hierarchies, you'll simply be shown all of the data members in the dimension.

The Slowly Changing Dimension (SCD)

The next consideration for you when modeling dimensional data is to review business requirements for the dimension data. You are looking for the desired outcome when dimension member data is updated or deleted. In OLAP modeling, inserting new dimension members is *not* considered a change. The only two cases you must be concerned with here are updates and deletes. This type of modeling is called slowly changing dimension (SCD) modeling.

The first question to ask of your SMEs is “What do you want to happen when dimension members no longer have fact data associated with them?” In our experience, most clients prefer to have dimension members marked as “not active” at this point, rather than deleted. In some cases, it has been a business requirement to add the date of deactivation as well.

The next case to consider is the business requirements for dimension member value updates. The most common scenario is names of people, that is, customers, employees, and so on. The question to ask here is “What do you want to happen when an employee changes his or her name (for example, women getting married)?”

Table 2–2 shows the four different possibilities in modeling, depending on the answer to the preceding question.

Table 2–2. Possible SCD Modeling Options

| Requirement | Description |
|---------------------|--|
| Last change wins | Overwrite with any change; do not retain original value. |
| Retain some history | Retain a fixed number of previous values. |
| Retain all history | Retain all previous values. |

Type 1, 2, 3 SCD Solutions

The table of requirements for dimension member changes (Table 2–2) is the basis for you to model the source tables using standard SCD type behavior modeling. This standard is implemented across a broad variety of OLAP products, including SSIS packages for ETL. In this case, we are thinking about SSIS packages that implement updates and deletes to dimension values.

Review the requirements and note that some dimension members will allow changes. You will need to translate those requirements to one of these standard solutions.

- *Type 1* means *overwriting* previous dimension member values, which is sometimes also called *last change wins*. This type is called a Changing Attribute in the SSIS Slowly Changing Dimension wizard.
- *Type 2* means adding a *new record* (or row value) when the dimension member value changes. This type is called a Historical Attribute in the SSIS Slowly Changing Dimension wizard.
- *Type 3* means adding additional *attributes* (or column values) when the dimension member value changes. This type is not supported in the SSIS Slowly Changing Dimension wizard.

Another important reason to use this standard modeling approach is that SSIS (and several other OLAP products) supports these terms and concepts in the new SCD data flow transformation task object. This is important because, although you’ll probably manually process updates or deletes to dimensional attribute member values during development, after you move to production, you’ll want to automate this process. Using the new SSIS transformation (with its associated configuration wizard) is a quick and easy way to move this type of process to a SSIS package. The key configuration wizard page for this SSIS transformation is shown in Figure 2–14. SSIS packages are designed to automate ETL processes across your BI project.

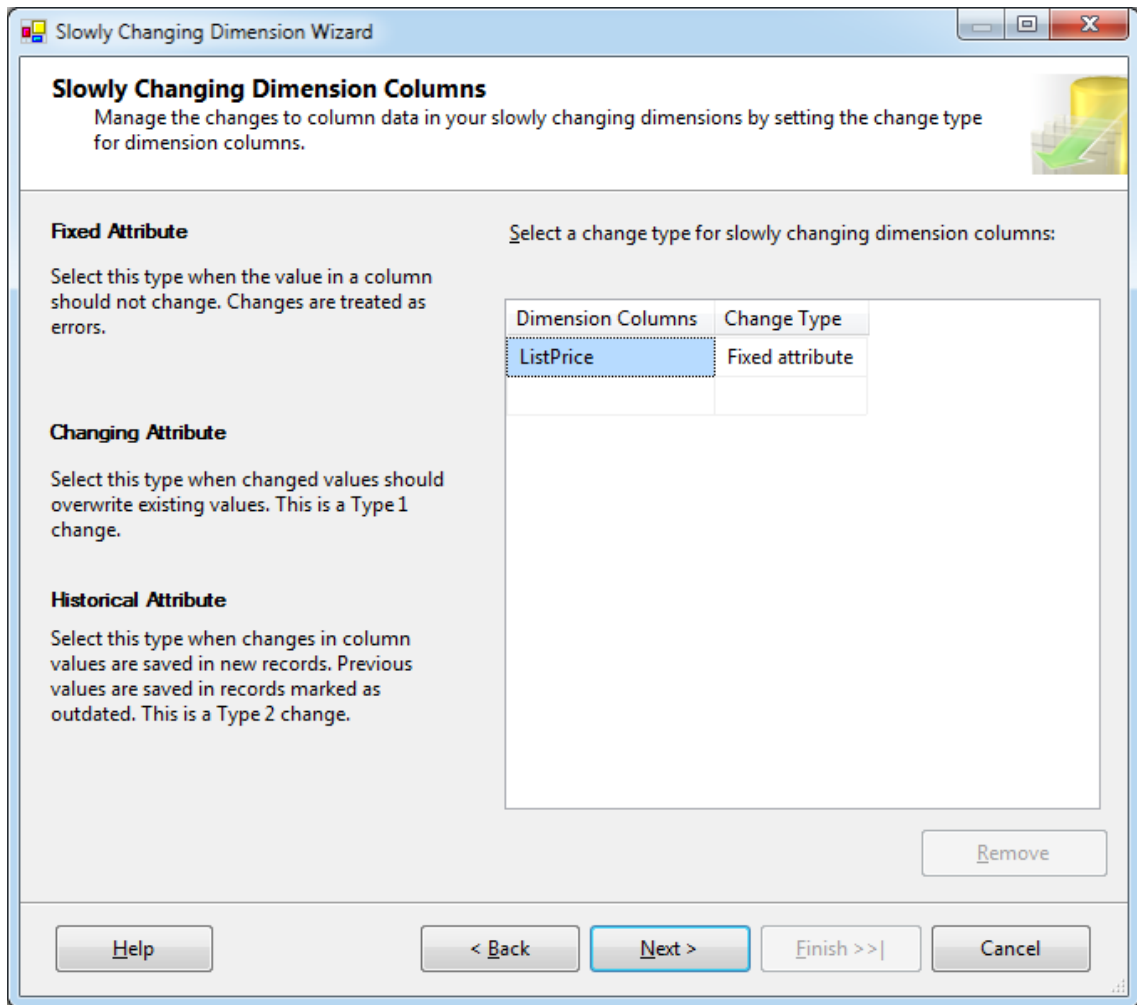


Figure 2–14. The Slowly Changing Dimension Wizard in SSIS helps you manage the data in this type of dimension.

The Rapidly Changing Dimension (RCD)

The rapidly changing dimension (RCD) is a dimension whose member values change constantly. *Constantly* is the operative word here. This should be a very small subset of your dimensional data. To work with this type of dimension, you will probably vary the storage location, rather than implementing any particular design in the OLAP model itself.

RCD data storage models are covered in more detail in Chapter 6, “Cube Storage and Aggregation” (with the rest of the discussion about overall cube data storage). An example of this type of dimension is

in a fast food restaurant chain, where the employee dimension may need to reflect very high staff turnover. The employee dimension is modeled as an RCD.

Writeback Dimension

Another advanced capability of dimensions is writeback. *Writeback* is the ability for authorized end users to update the data members in a dimension: insert, update, or delete. In our experience, only a very small number of business scenarios warrant enabling writeback for particular cube dimensions. If you are considering enabling writeback, verify that it is acceptable given any regulatory requirements, such as SOX, HIPAA, and so on, in your particular business environment.

There are some restrictions if you want to enable writeback. The first restriction is that the dimension must be based on a single table, meaning it must use a star schema modeling format. The second restriction is that writeback dimensions are only supported in the Enterprise Edition of Analysis Services. Finally, writeback security must be specifically enabled at the user level. This is covered in more detail in Chapter 15, “SSAS Administration.”

Understanding Fact (Measure) Modeling

One key and core part of your BI solution is the business facts that you choose to include. As mentioned previously, facts are also called measures. *Measures* are the key metrics by which you ascertain the success of your business. Some examples include daily sales amount, product sales quantity, net profit, and so on. Clearly selecting the appropriate facts is a critical consideration in your model.

An Example

If you have been thorough in the business requirements gathering phase of your BI project, modeling facts should be simple. In most cases, facts are numeric and are aggregated by summing the facts across all levels of all dimensions. There are, however, exceptions to this rule. An example of the most typical case is a cube that captures sales activity as shown in Figure 2–15 from the AdventureWorks 2008 R2 sample.

Figure 2–15 shows all of the measures, the measure group (which is simply a folder that has the same name as the source fact table), the data type, and the aggregation type from the cube design tab in BIDS. You may notice that several measures use Sum as their aggregation type. There are two Count measures shown in the figure. Sum is the default type of aggregation in SSAS.

The built-in aggregation types available for use in SSAS are listed in Table 2–3. Note that the table lists the Type of measure. This is a descriptor of the aggregation behavior. *Additive* means to roll up the one ultimate total. *Semiadditive* means to roll up to a total for each level, but not cumulatively, and is applicable only over a time dimension. *Nonadditive* means to *not* roll up, that is, only shows that particular value. Also note that semiadditive measures require the Enterprise Edition of SSAS.
















| Measures | | | | |
|---|------------------------------|--------------------|-----------|---------------|
| | Name | Measure Group | Data Type | Aggregation |
|  | Internet Sales Amount | Internet Sales | Currency | Sum |
|  | Internet Order Quantity | Internet Sales | Integer | Sum |
|  | Internet Extended Amount | Internet Sales | Currency | Sum |
|  | Internet Tax Amount | Internet Sales | Currency | Sum |
|  | Internet Freight Cost | Internet Sales | Currency | Sum |
|  | Internet Unit Price | Internet Sales | Currency | Sum |
|  | Internet Total Product Cost | Internet Sales | Currency | Sum |
|  | Internet Standard Product... | Internet Sales | Currency | Sum |
|  | Internet Transaction Count | Internet Sales | Integer | Count |
|  | Internet Order Count | Internet Orders | WChar | DistinctCount |
|  | Customer Count | Internet Customers | Integer | DistinctCount |
|  | Sales Reason Count | Sales Reasons | Integer | Count |
|  | Reseller Sales Amount | Reseller Sales | Currency | Sum |
|  | Reseller Order Quantity | Reseller Sales | Integer | Sum |
|  | Reseller Extended Amount | Reseller Sales | Currency | Sum |

Figure 2–15. The Measures window (grid view) in BIDS allows you to view and alter properties of measures used in your cube.

Table 2–3. Aggregation Functions Available in SSAS

| Aggregation | Type |
|-----------------------------|--------------|
| Sum | Additive |
| Count, Min, Max | Semiadditive |
| FirstChild, LastChild | Semiadditive |
| AverageOfChildren | Semiadditive |
| FirstNonEmpty, LastNonEmpty | Semiadditive |
| ByAccount | Semiadditive |
| DistinctCount, None | Nonadditive |

■ **Note** ByAccount aggregation refers to an aggregation that calculates according to the aggregation function assigned to the account type for a member in an account dimension. An *account dimension* is simply a dimension that is derived from a single, relational table with an account column. The data value in this column is used by SSAS to map the types of accounts to well-known account types (for example, assets, balances, and so on) so that you replicate the functionality of a balance sheet in your cube. SSAS uses these mappings to apply the appropriate aggregation functions to the accounts. If no account type dimension exists in the measure group, ByAccount is treated as the None aggregation function.

Calculated Measure vs. Derived Measure

A final consideration for measures is that you can elect to derive measure values on load of data into the cube. This type of measure is called a *derived measure* because it is derived, or created, when the cube is loaded, rather than simply retrieved using a SELECT statement from the source fact table(s). Creating derived measures is done via a statement (T-SQL for SQL Server) that is understood by the source database. We do not advocate using derived measures because the overhead of creating them slows cube processing times.

Rather than incurring the overhead of deriving measures during cube loads, an alternative way to create the measure value is to calculate and store the measure value during the ETL process, which is used to load the (relational) fact table, rather than in the SSAS cube. That way, the value can simply be retrieved (rather than calculated) during the cube load process.

In addition to derived measures, SSAS supports calculated measures. *Calculated measure* values are calculated at query time by SSAS. Calculated measures execute based on queries that you write against the OLAP cube data. These queries are written in the language required for querying SSAS cubes, which is Multidimensional Expressions (MDX). We will review the process for creating calculated measures in Chapter 13, “Introduction to MDX.”

Other Types of Modeling

SSAS supports additional capabilities that may affect the final modeling of your cube source schemas. In our experience, most clients start with the concepts presented in this chapter, load some sample data to validate both the data and the modeling concepts, and then add the more advanced capabilities.

Data Mining

Data mining capabilities have been enhanced with each version of SSAS, due to the growing number and increasing sophistication of data mining algorithms. *Data mining* is the ability to use predefined algorithms to detect patterns in the data. Interestingly, SSAS’s data mining capabilities can be used with either OLTP or OLAP source data. Data mining is covered in more detail in Chapter 14, “Introduction to Data Mining.”

Key Performance Indicators

A *key performance indicator (KPI)* is a method (usually displayed in an end-user tool visually) of showing one or more key business metrics: the current state, comparison to goal, trend over time, and other information. KPIs are usually shown via graphics, such as, red, yellow, or green traffic lights; up arrows or down arrows; and so on. You'll learn about the planning and implementation of SSAS KPIs in Chapter 4, "Intermediate OLAP Modeling with SSAS."

Actions, Perspectives, Translations

SSAS *actions* give the end users the ability to right-click a cell of the cube and to perform some type of defined action, such as passing the value of the selected cell into an external application as a parameter value and then launching that application. *Perspectives* are similar conceptually to relational views. They allow you to create named subsets of your cube data for the convenience of your end users. *Translations* give OLAP modelers a quick and easy way to present localized cube metadata to end users. All of these capabilities are also covered in Chapter 4, "Intermediate OLAP Modeling with SSAS."

Source Control and Other Documentation Standards

Already in the OLAP modeling phase, your BI project will contain many files of many different types. While you are in the modeling phase, the files will probably mostly consist of Visio diagrams, Excel spreadsheets, and Word documents. It is very important to establish a methodology for versioning and source control early in your project. When you move to the prototyping and developing phase, the number and types of files will increase exponentially.

You can use any tool that works for you and your team: Rational ClearCase, Perforce, Visual Source Safe (VSS), Visual Studio Team System, SharePoint Document Libraries, or versioning via Office. The important point is that you must establish a system that all of your BI team members are committed to using early in your BI project lifecycle. Also it's important to use the right tool for the right job; for example, SharePoint Document Libraries are designed to support versioning of requirements documents (which are typically written using Word, Excel, and so on), whereas VSS is designed to support source control for OLAP code files, which you'll create later in your project's lifecycle.

Another important consideration is naming conventions. Unlike OLTP (or relational) database design, there are very few common naming standards in the world of OLAP design. I suggest that you author, publish, and distribute written naming guidelines to all members of your BI team during the requirements gathering phase of your project. These naming guidelines should include suggested formats for the following items at minimum: cubes, dimensions, levels, attributes, star schema fact and dimension tables, SSIS packages, SSRS reports, SharePoint pages, and dashboards.

Summary

This chapter covered the basic modeling concepts and techniques for cubes in a BI project. You saw how grain statements can be used for a high-level validation of your modeling work. You learned how best to determine what types of dimensions (fixed, SCD, or RCD) and facts (stored, calculated, or derived) will be the basis for your cubes. We also discussed the concept of hierarchies of dimensional information. If you are new to BI, you've got some "unlearning" to do. OLAP modeling is very dissimilar to OLTP modeling, mostly because of the all-prevalent concept in OLAP of deliberate denormalization.

In the next chapter, we'll introduce you to OLAP modeling with SSAS. In addition, you will discover more about measures and dimensions. Finally, you will build your first cube!



Introducing OLAP Modeling with SSAS

Now that you've been introduced to OLAP modeling concepts, let's take a look at OLAP modeling with SQL Server Analysis Services (SSAS). In this chapter, you will continue to discover OLAP modeling concepts and techniques using SSAS, and you will build your first cube. It is quite common to prototype cubes built on subsets of enterprise data quickly in a BI project. As with any other type of development, you can expect cube development to be iterative. Generally, extract, transform, and load (ETL) development runs somewhat concurrently to these cube iterations, assuming that you have the resources to commit to both of these processes.

This chapter assumes you have a couple of populated star schemas in SQL Server to work with. The data in the samples are, of course, very clean. This is not to ignore the real-world situation of data cleansing, validation, and transformation; rather, it allows you to focus on cube building using the many features available in SSAS. So, you'll use the handy AdventureWorks 2008R2 sample that is part of the SQL Server sample databases as a source for building your first cube. If you haven't yet installed the sample database, refer to the explanation in "Building the First Sample—Using AdventureWorks" in Chapter 1. This chapter will cover the following topics:

- Using SSAS in BIDS, and understanding the development environment
- Creating data sources and Data Source View objects
- Creating cubes using the UDM and the Cube Build Wizard
- Refining dimensions and measures in BIDS

Using BIDS to Build a Cube

In this section, you will use the Business Intelligence Development Studio (BIDS) to create SSAS cubes. This environment is also used to create SSAS data mining structures and models (which are covered in Chapter 14) and SQL Server Reporting Services reports and report models (covered in Chapter 10). To start your work, open BIDS, and select File ► New ► Project. Select the Analysis Services Project template under the "Visual Studio installed templates" heading, as shown in Figure 3-1.

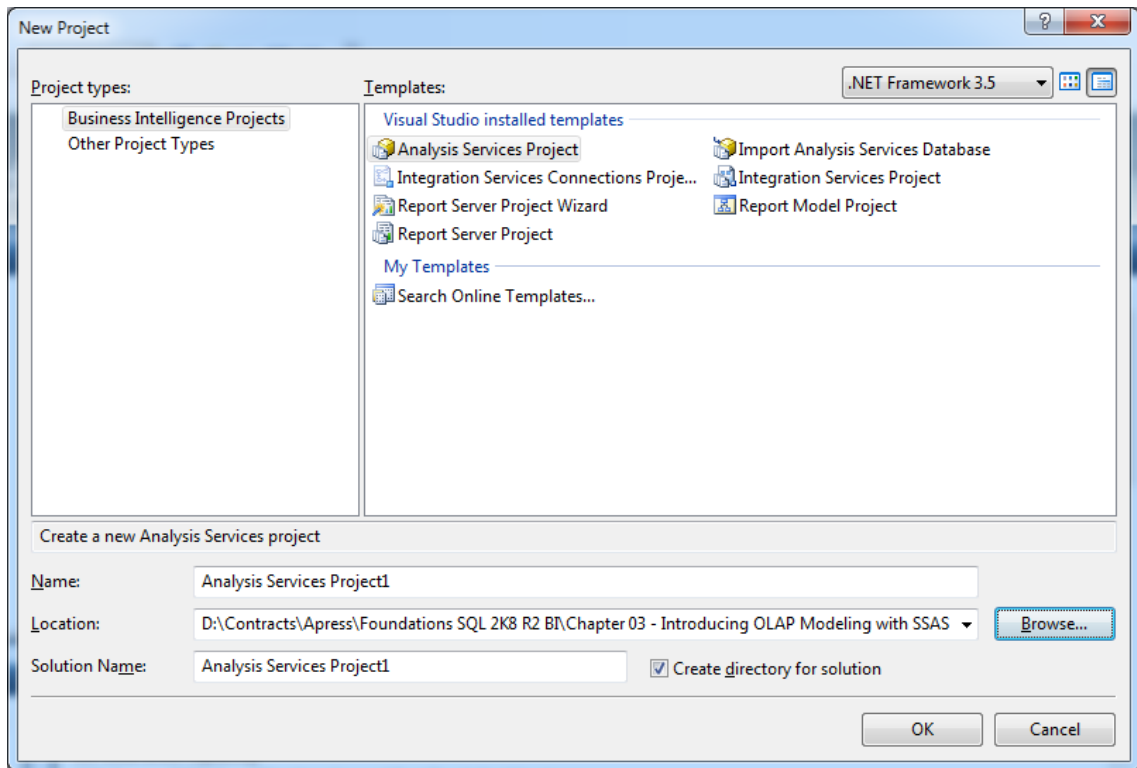


Figure 3–1. The New Project dialog box in BIDS allows you to select project templates to create a new Analysis Services solution or to import an SSAS database.

In Solution Explorer, you'll note several new folders or nodes. Each node is designed to hold a different type of item or file. The first node, Data Sources, will contain connections that can be used in multiple packages. A data source stores server, database, and security information. Data Source Views (DSVs), shown in the second node, are defined against one of your data sources. These views, which are analogous to database views, allow you to create calculated columns and define relationships between tables in the DSV. DSVs are an important feature for SSAS designers who want to make usability improvements against the star schema. In some situations, SSAS designers will not have permissions to create objects (such as views) in source star schema databases.

Some of the enhancements that can be made via DSVs are as follows:

- Rename tables or columns to create more end-user friendly names.
- Add calculated columns, which can include column concatenations, or other manipulations that the source database understands (in our case, using T-SQL).
- Remove columns that are not needed for the UDM.
- Add derived measures to the fact table, much like calculated columns for the dimension tables (in our case using T-SQL).

To create a data source, right-click the Data Sources folder in Solution Explorer, and select New Data Source. Click Next in the Welcome dialog. On the “Select how to define the connection” dialog, click New, and the Connection Manager will be displayed. In the Connection Manager, enter your server name and authentication method. Choose AdventureWorksDW2008R2 as the database to connect to. The completed dialog should resemble Figure 3–2.

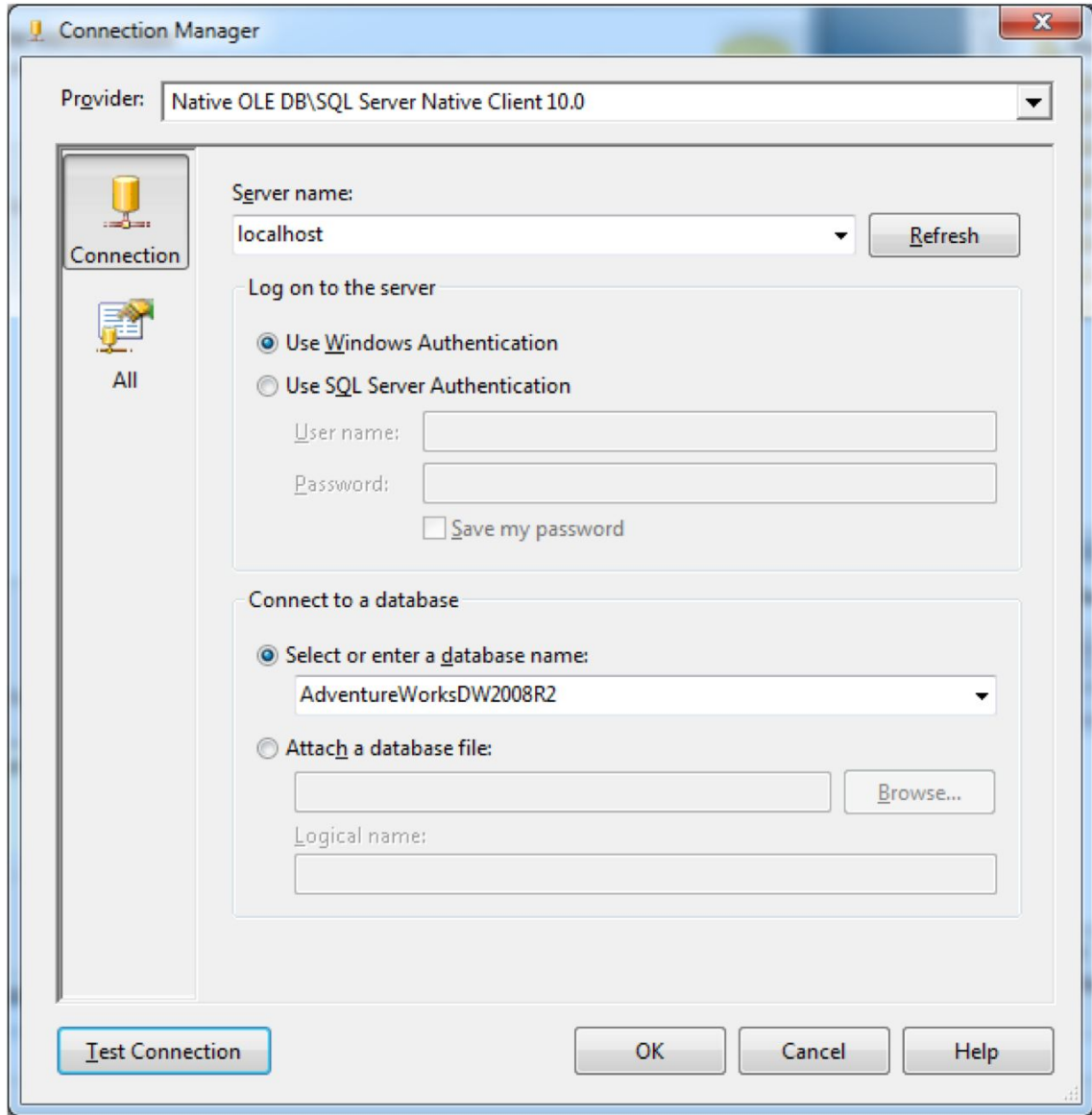


Figure 3–2. Use the Connection Manager to set the properties of a connection.

Once you have completed setting your connection's properties, click OK. The final dialog in this wizard will display your data connection's name, and the properties you set. Click the Finish button to confirm your settings. Finally, complete the wizard by naming your data source Adventure Works DW2008R2 and clicking Finish. Figure 3–3 shows the confirmation dialog.

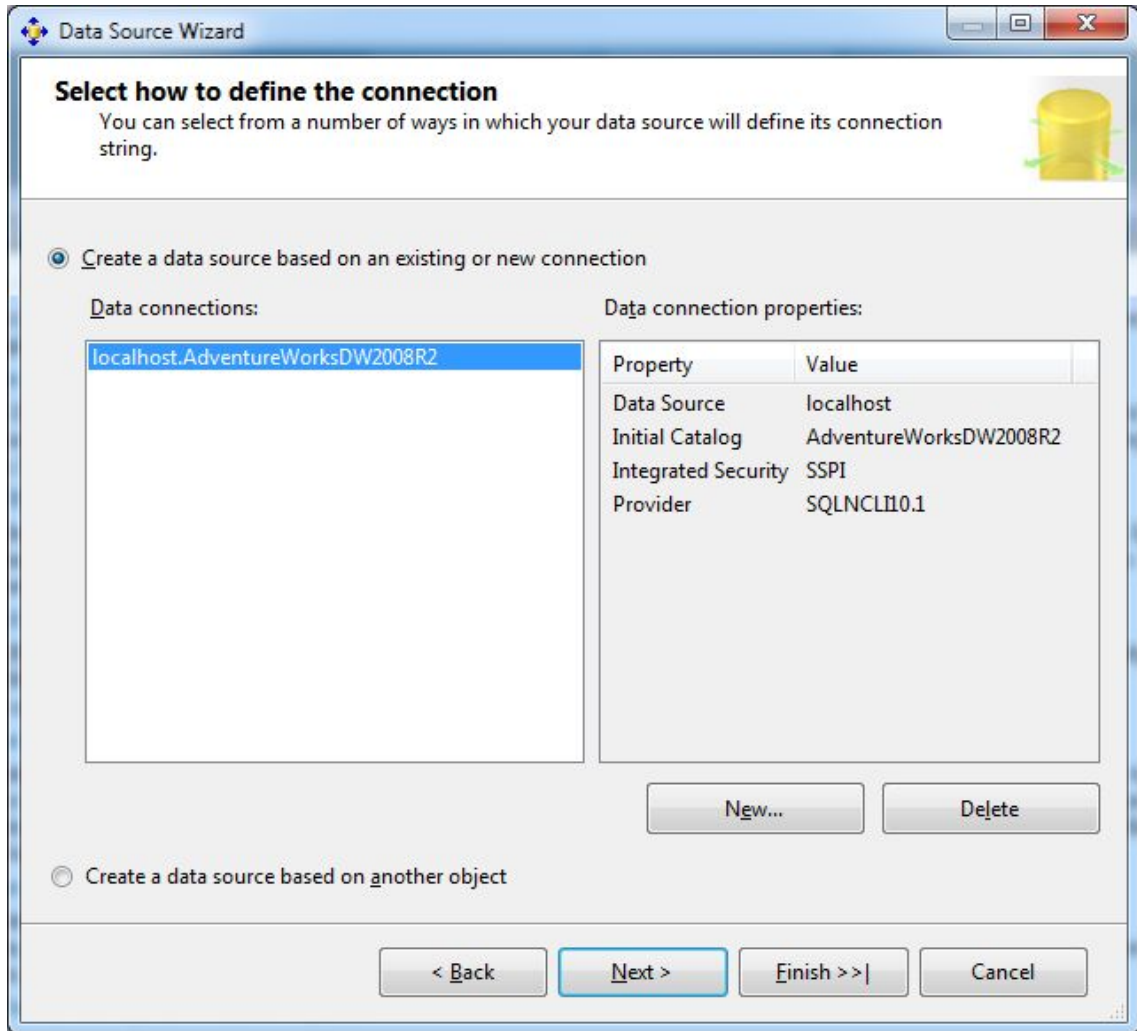


Figure 3–3. Use this dialog of the Data Source Wizard to confirm your connection settings.

Creating a DSV is similar to creating a data source. To create a DSV, right-click the Data Source Views folder in Solution Explorer, and select New Data Source View. Click Next in the Welcome dialog. The Select a Data Source dialog will contain the data source you just created. Select this data source, and click Next. In the Select Tables and Views dialog, select the following tables from the “Available objects”

area, and place them into the Included objects: area, using the right-facing arrow button or by drag-and-drop:

- DimCustomer
- DimDate
- DimEmployee
- DimGeography
- DimProduct
- DimProductCategory
- DimProductSubcategory
- FactInternetSales
- FactResellerSales

Figure 3–4 displays the nine tables that you will be using to create your first cube. When you are finished making your table choices, click Next. Complete the wizard by naming your data source view Adventure Works DW2008R2 and clicking Finish.

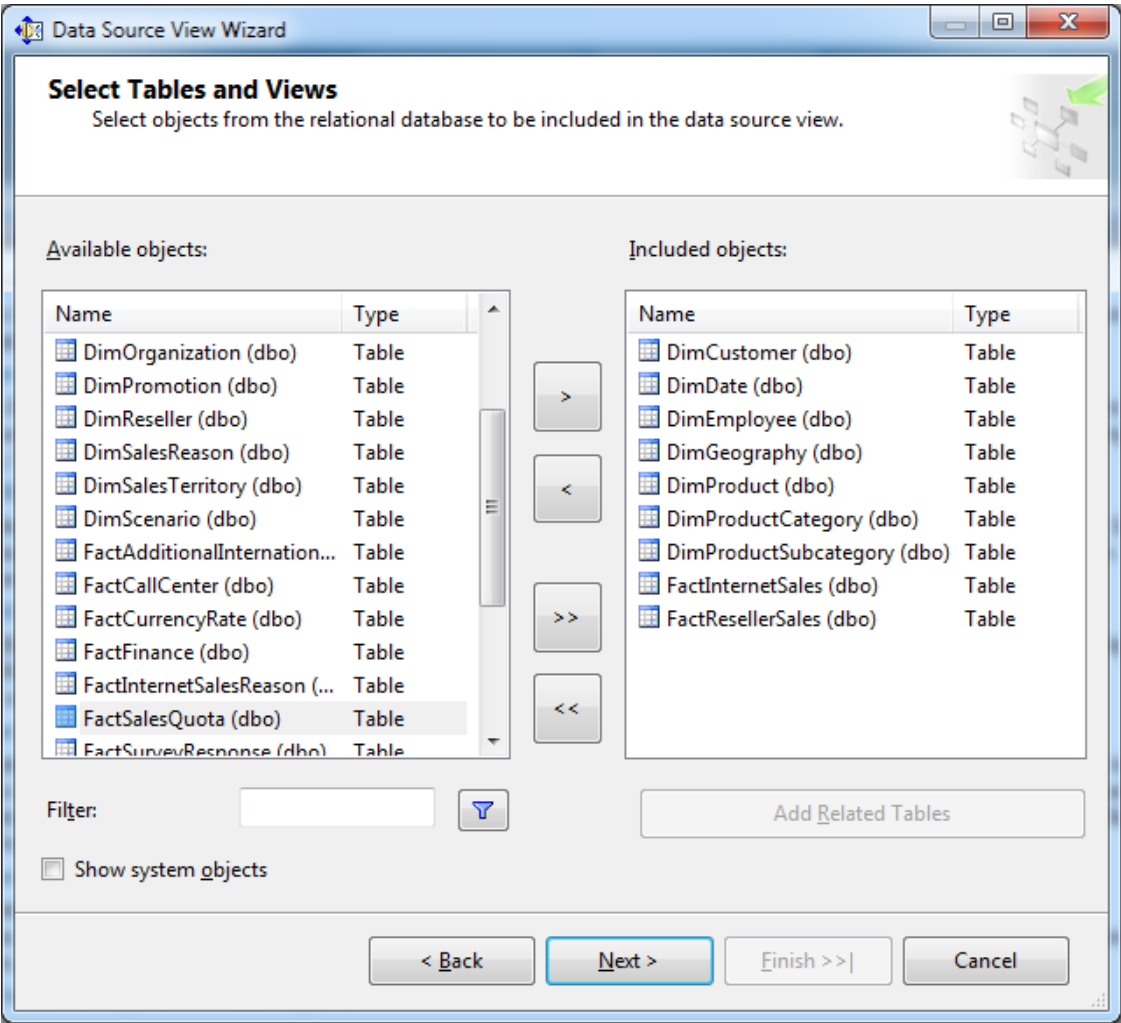


Figure 3–4. Use the *Select Tables and Views* dialog to include tables and views in your data source view.

After some processing, BIDS will display your tables and their relationships as an entity relationship diagram (ERD) in the main area of the Design tab. Figure 3–5 shows the Design tab, with your tables displayed in the main area.

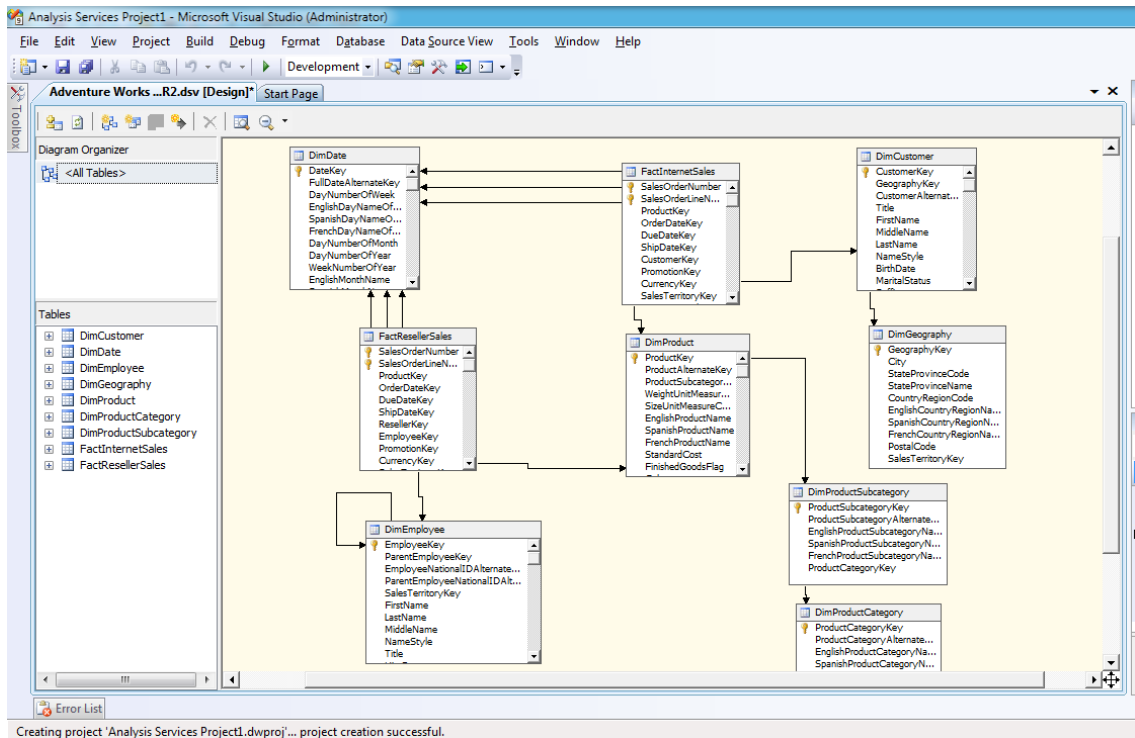


Figure 3–5. Use the Design tab to view the tables that belong to your data source view. You'll use the DSV to build your cube.

You can explore the data in the tables and views in your DSV by right-clicking any table in the DSV window and clicking **Explore Data**. You'll then be presented with four different ways to look at the source data: via a table, a pivot table, a chart, or a pivot chart. This feature can be helpful in the development of a useful DSV as a basis for your cube, because it allows you to easily and quickly explore the data in a variety of output formats. You can quickly validate data values by examining the data in tabular and charted output formats.

To interact with your employee source data using the familiar Office PivotTable interface, explore **Employee**, and click the **Pivot Table** tab. Use the PivotTable Field List to add **Marital Status** to the Row Field area, then add **Status** to the Column Field area. Next, drop **BaseRate** in the Totals or Detail Fields area. Finally, add **Gender** to the Filter Field area, and filter on the value **F**. Figure 3–6 shows the pivot table option for the **DimEmployee** table.

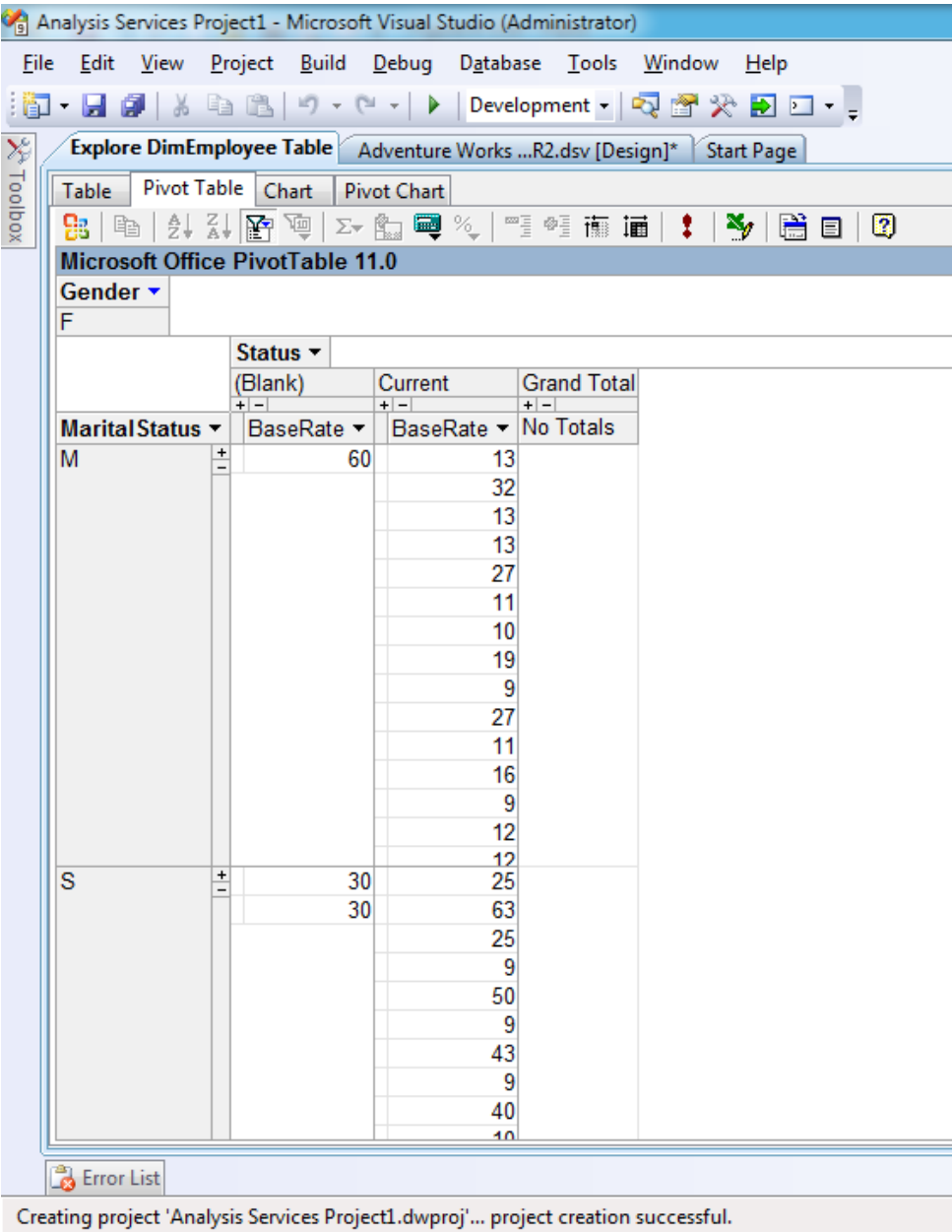


Figure 3–6. Using the Explore Data feature of the DSV in BIDS can help you make intelligent decisions about refining the DSV for your cube.

NAME MATCHING

An interesting option available in the DSV is displayed only if foreign key (FK) relationships do not exist in the underlying data source. This is the `NameMatchingCriteria` property, which allows you to specify whether you prefer to have Analysis Services generate relationships between columns in the source tables of the DSV automatically.

- Same name as primary key
- Same name as destination table name
- Destination table name and primary key name

The three options to pick from are listed:

Defining Your First Cube

After creating at least one data source and at least one DSV, you are ready to define your first cube. The Cube Wizard will help you define cubes quickly and correctly. You can (and usually will) make many refinements to OLAP cubes generated using the wizard after the initial run of the wizard.

Also, this flexible and useful wizard includes the capability to go back at any point, so if you are not happy with the results of a particular page, you can click Back, adjust input parameters, and then proceed again. It is also the only way to create a new cube in BIDS. To access the Cube Wizard, right-click the Cubes folder in Solution Explorer, and click New Cube. Click Next in the Welcome dialog. Choose “Use existing tables” as your creation method, and click Next. The wizard will now ask you to select your measures; choose `FactInternetSales` and `cFatResellerSales`, and click Next.

The Select Measures dialog is where you choose which facts you want to aggregate in your cube. As it is not meaningful to aggregate your primary or foreign keys, deselect each of these. `Revision Number` is an example of a nonkey fact that is an attribute and not a measure. Deselect `Revision Number` and `Revision Number - Fact Reseller Sales` as well. For your first cube, your selected measures should look like Figure 3–7.

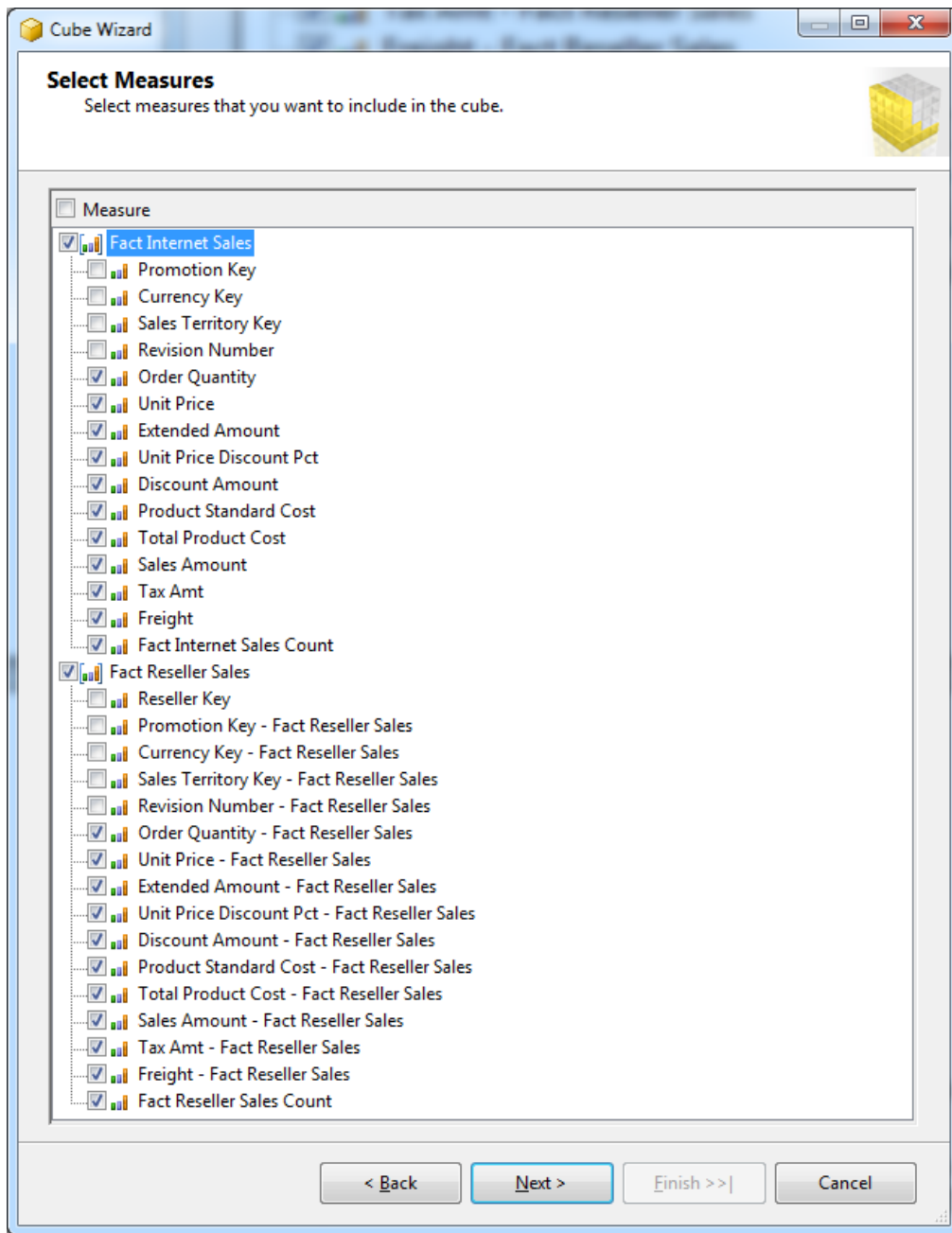


Figure 3–7. Use the Select Measures dialog to choose the measures that belong in your cube.

Once you have chosen your measures, click Next to continue. On the Select New Dimensions dialog that appears next, deselect Fact Internet Sales and Fact Reseller Sales. Click Next to continue. Finally, review your selections in the preview area, name your cube Adventure Works DW2008R2, and click Finish to complete the Cube Wizard. When the wizard completes, the Cube Structure tab will open, displaying your data source view, your measures and dimensions. The Cube Structure tab for your newly created cube is shown in Figure 3–8.

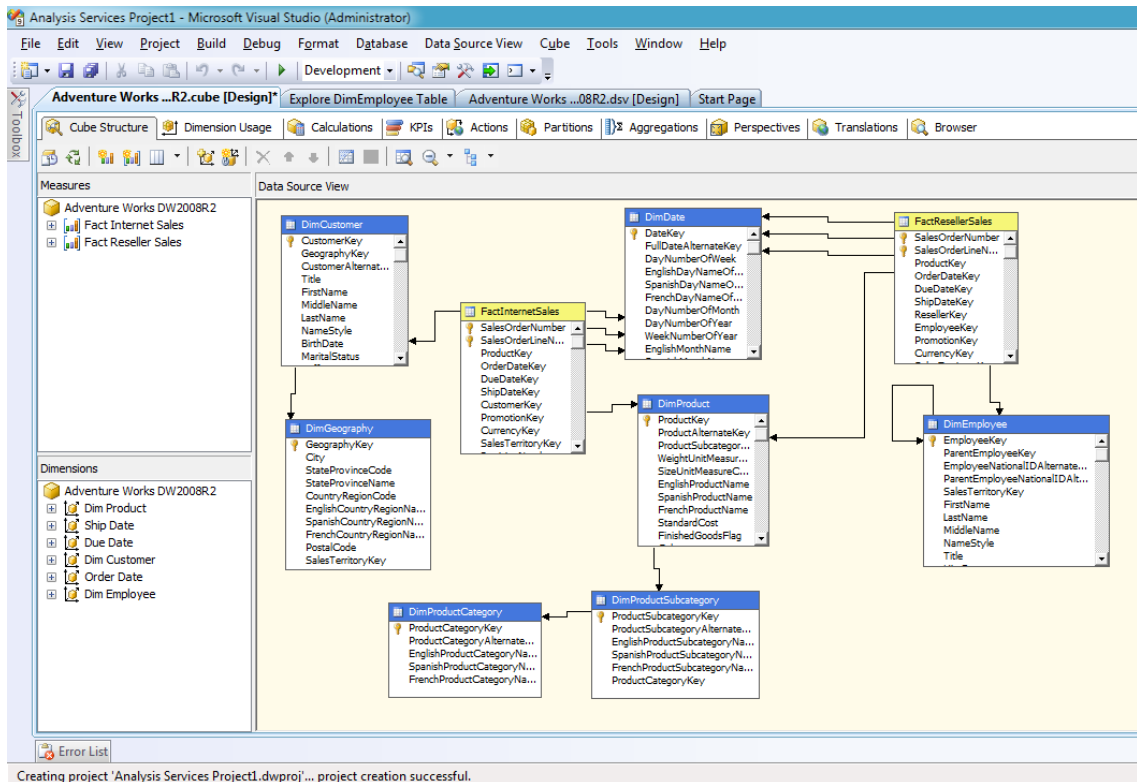


Figure 3–8. The Cube Structure tab shows the data source view, as well as the measures and dimensions for a cube.

Table 3–1 lists each of the steps you performed in the Cube Wizard, the information presented by BIDS, and the action that you took.

Table 3–1. Steps you performed using the Cube Wizard in BIDS

| Wizard Step | Actions by BIDS | Actions by You |
|-----------------------------|---|--|
| Select creation method | -- | Select the “Use existing tables” radio button. |
| Select measure group tables | Detects and displays data source views and measure group tables | Select Adventure Works DW2008R2 and each fact table. |
| Select measures | Detects and displays candidate measures from the Fact tables you selected in the preceding step | Deselect attributes that will not be aggregated. |
| Select new dimensions | Detects and displays candidate dimension tables | Deselect the fact tables. |
| Completing the wizard | Displays a suggested name for the cube and its contents | Name the cube Adventure Works DW2008R2. |

Adding Dimension Attributes

Now that your cube is defined, it is time to work with your dimensions by adding attributes. In this section, you will use BIDS to add attributes to the Date dimension. To begin, open the dimension designer for the Date dimension by double-clicking Dim Date in Dimensions folder of your project. Next, drag and drop the CalendarYear and MonthNumberOfYear fields from the Data Source View pane to the Attributes pane. Figure 3–9 shows the Dimension Structure tab, with the Date dimension and the attributes you have chosen.

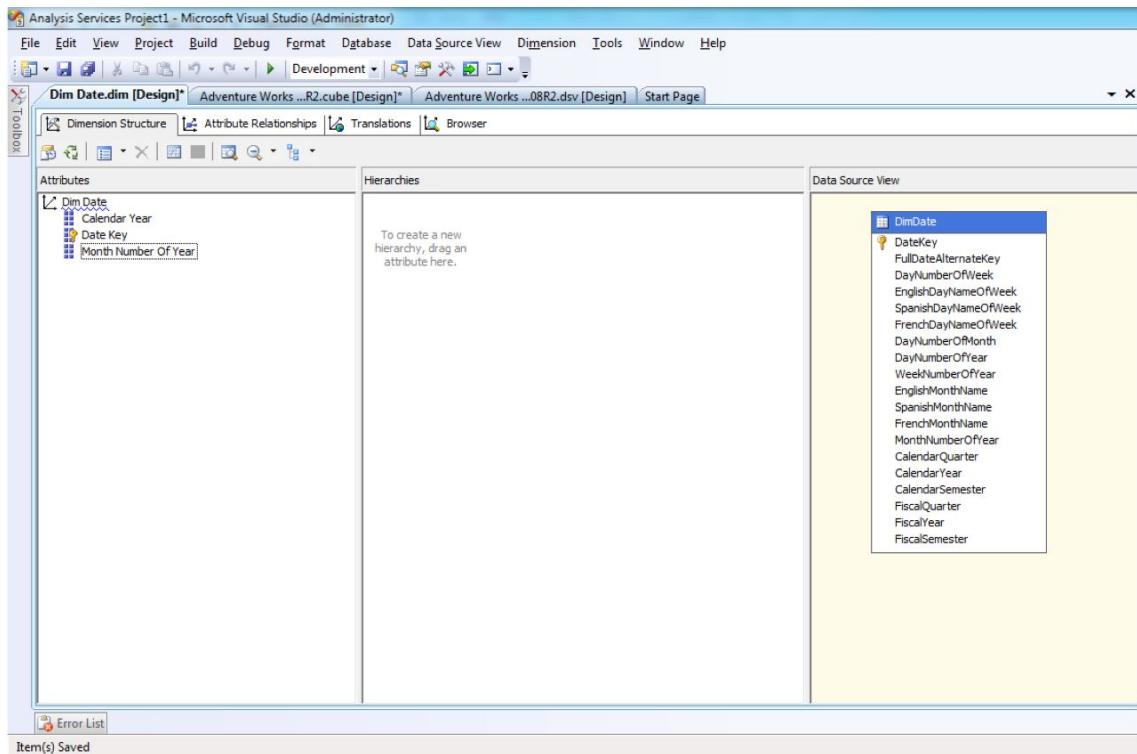


Figure 3–9. The Dimension Structure tab, showing the Date dimension and your chosen attributes

Now that you have created the date attributes for your first cube, let's move on to the Product dimension. Open the Product dimension in the dimension designer, and take note of the three tables in the data source view. You will use these very soon to create your first hierarchy. For now, add the following fields to the Attributes pane: Color, EnglishProductCategoryName, EnglishProductSubcategoryName, and EnglishProductName. Figure 3–10 shows the Product dimension with four added attributes.

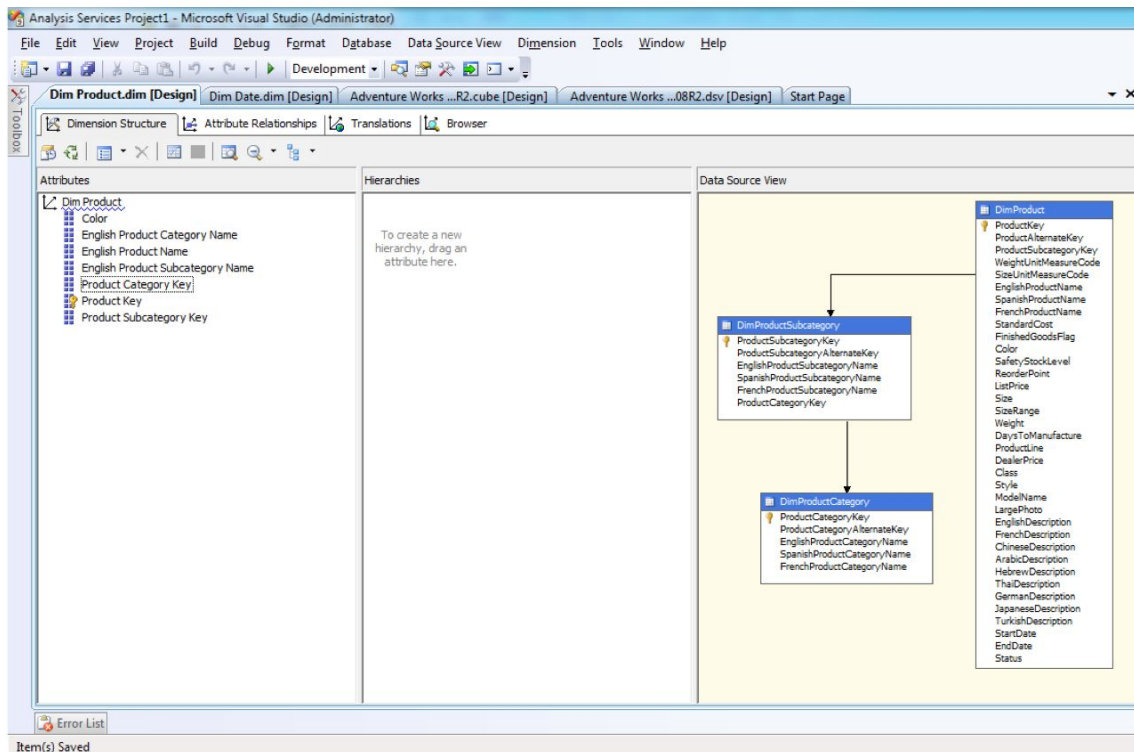


Figure 3–10. The Dimension Structure tab, showing the Product dimension tables and your chosen attributes

Defining Hierarchies

Hierarchies play an important role in your cube development. They add usability to your cube from an end-user perspective—they enable users to navigate and filter the cube. The Product dimension is an example of a *natural hierarchy*. It is easy to visualize that a product belongs to a product subcategory, which, in turn, belongs to a product category.

To create the product hierarchy, open the Product dimension in the dimension designer. Begin by dragging the English Product Category Name attribute into the Hierarchies pane. When you do this, BIDS will create a hierarchy object, with English Product Category Name highlighted and an extra row named <new level>. Next, drag and drop the English Product Subcategory Name attribute to the <new level> row, followed by the English Product Name attribute. Figure 3–11 shows the completed Product dimension. You can safely ignore the warning icon displayed in the hierarchy for now. The “Creating Attribute Relationships” section later in this chapter will cover this warning in more detail.

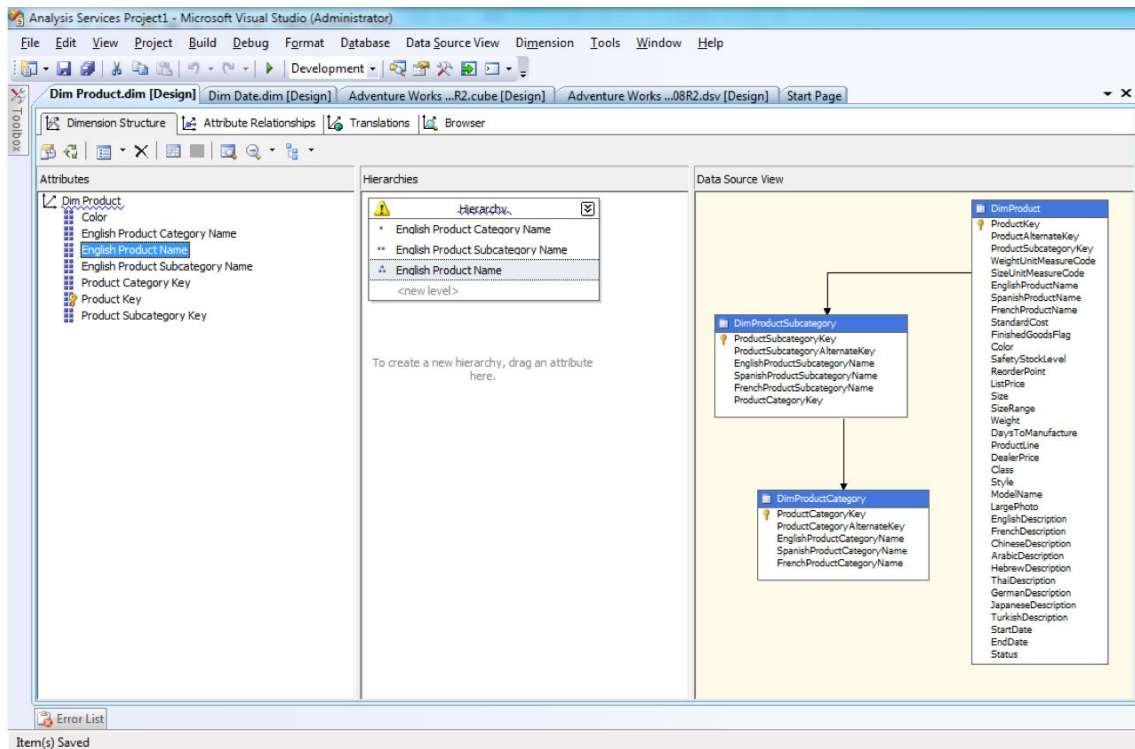


Figure 3–11. The Dimension Structure tab, showing the Product dimension attributes and hierarchy

Building Your First Cube

Let's review what you've completed in preparation for understanding how you can further edit and refine your cube in BIDS. To browse your cube, you'll have to build and deploy it first. During the build step, the cube metadata is validated (if there are errors in the metadata, they will be reported to you and the cube will fail to build), and then it is written (called "deployed" in this interface) to the Analysis Services directories so that it can be used by query tools. The tool in BIDS for querying the cube is called the Cube Browser. Until you build and deploy your cube, you will not be able to view the cube results in BIDS. To build the cube you just created (which makes the metadata files that you've generated in BIDS available to SSAS), in the Solution Explorer view, right-click the cube, and click Process. BIDS will post a message that the server content appears to be out of date. Click Yes to build and deploy this project for the first time. The Process Cube dialog box appears, as shown in Figure 3–12. The details of processing are covered in Chapter 6, so for now, just click Run.

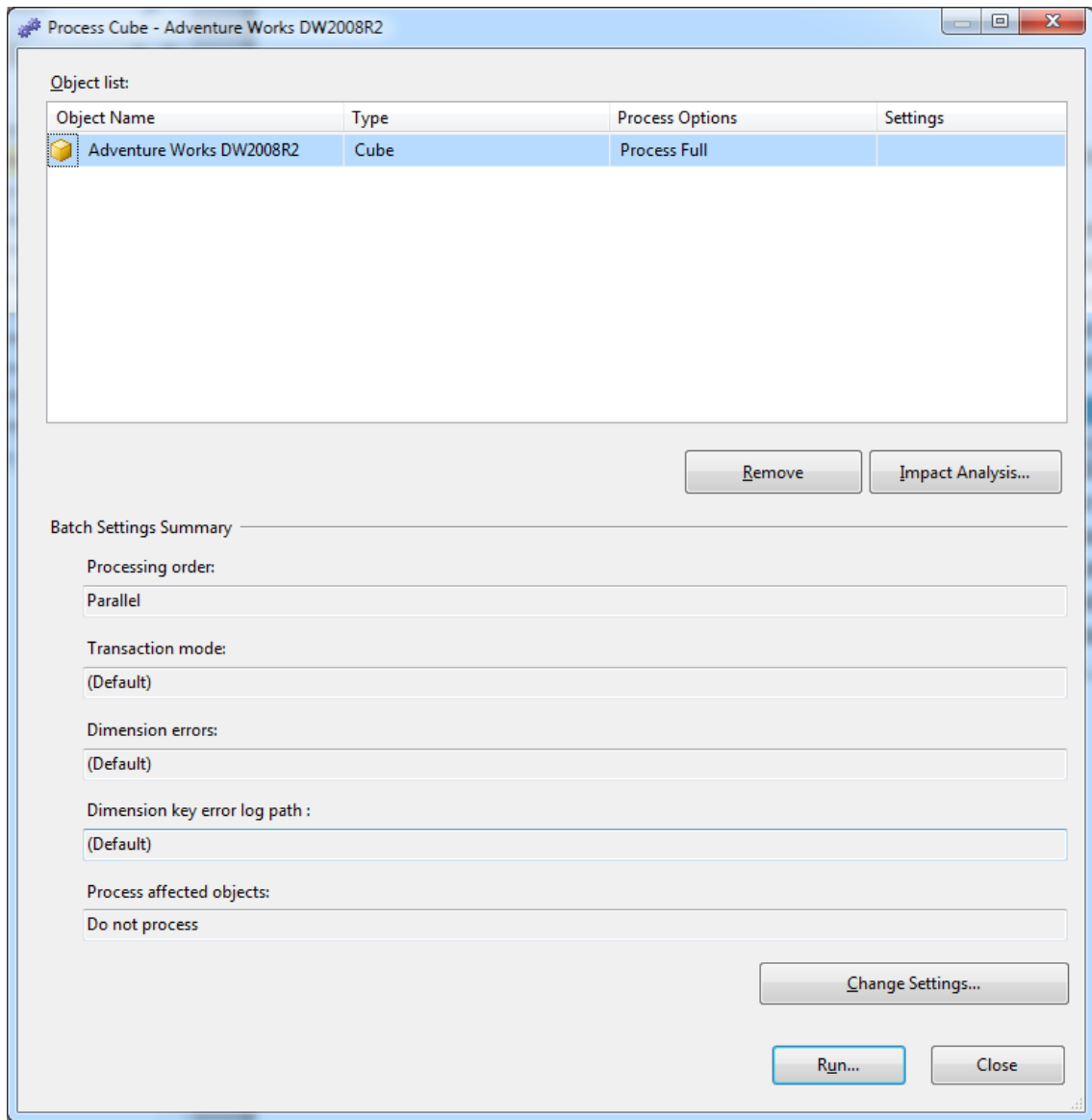


Figure 3–12. To browse your cube in the BIDS cube browser, you must first process and deploy it.

After you’ve clicked Run, BIDS shows a detailed cube-processing dialog box, which shows each step and the progress status. You are looking for a “Process succeeded.” message in the status area of this dialog box, as shown in Figure 3–13.

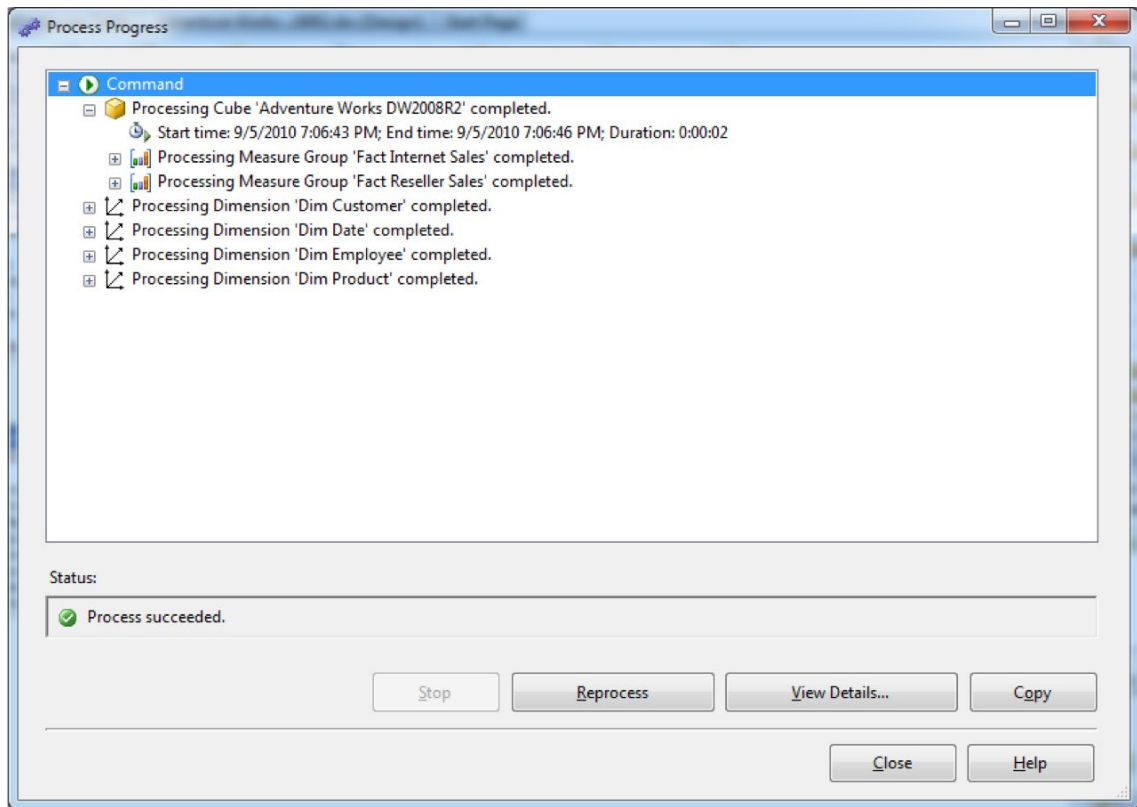


Figure 3–13. The Process Progress dialog box in BIDS displays detailed information about each step in the cube process task.

After you see this message, click Close to dismiss the Process Progress dialog; then click Close on the Process Cube dialog. To view your cube, select the Browser tab. This opens the familiar pivot table interface. Figure 3–14 displays the Sales Amount measure with the Calendar Year hierarchy on the rows pivot table axis, the Product Category hierarchy on the columns axis, and a filter to show only sales results for the All-Purpose Bike Stand and HL Road Tire products. Your browse results may look different from Figure 3–14, of course, depending on which values from the metadata browser you’ve chosen to add to the (cube) Browser area.

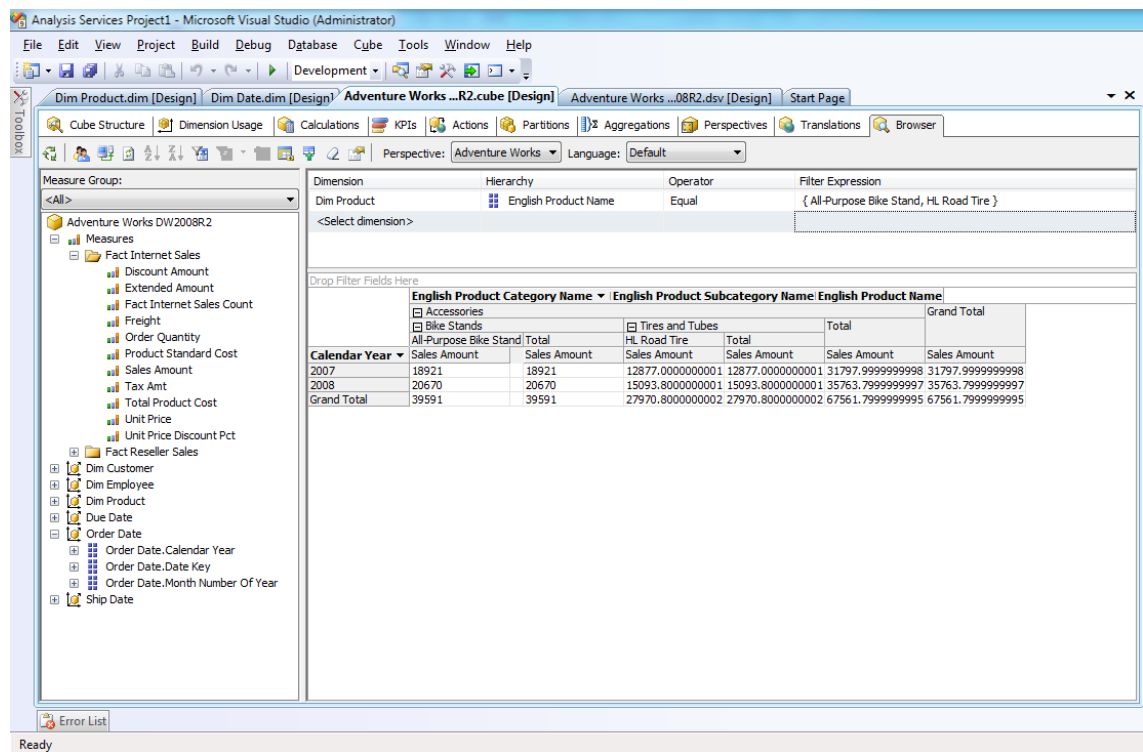


Figure 3–14. Browsing your first cube using BIDS, you can see the cube from an end-user perspective.

Refining Your Cube

You can do many, many tasks to refine a cube that you’ve built using the BIDS wizard. In this chapter, we’ll look at some simple and common cube refinements, including enhancing measures, dimensions, and the cube itself.

You’ll learn the BIDS development interface as you proceed with these refinements. Although the environment is a GUI, it is complex; often, half the battle when working with BIDS is getting to the correct location in the development interface!

Reviewing Measures

A common and simple task when refining your cube is changing the display format of the measures. To do this, you just double-click the cube name (under the cubes folder in Solution Explorer) to open that cube in the BIDS cube design interface. This will load the cube metadata into a series of tabbed interfaces, with the Cube Structure tab open by default. In the Measures panel on the left, right-click the measure that you want to format, and then click Properties. On the bottom right of the BIDS design surface, the property pane will be set to that particular measure. Click the selection list for the FormatString property, and select the format type that you need. The property view is shown in Figure 3–15. Among the values you can select:

- Standard
- Currency
- Percent
- Short Date
- Long Date
- 0
- #,##0.00;-#,##0.00
- \$#,##0.00;(\$#,##0.00)#,##0.00 %;-#,##0.00 %
- M/d/yyyy
- dddd, MMMM dd, yyyy

Another interesting use of measure properties in this same interface is to create a derived measure. A *derived measure* is based on a calculation that is “passed through” to the data source for the cube on process. In our case, this would be a T-SQL expression. You enter the expression in the MeasureExpression property in the Properties window for the measure you want to create.

You may choose to do this because the measure may not exist in the source data. Although this is a very flexible option to create additional measures, you should evaluate the overhead that creating a derived measure adds to cube-processing time. You can do this by reviewing the detailed output from the cube-processing window (discussed earlier in this chapter), paying particular attention to the amount of time taken to perform the derivation step.

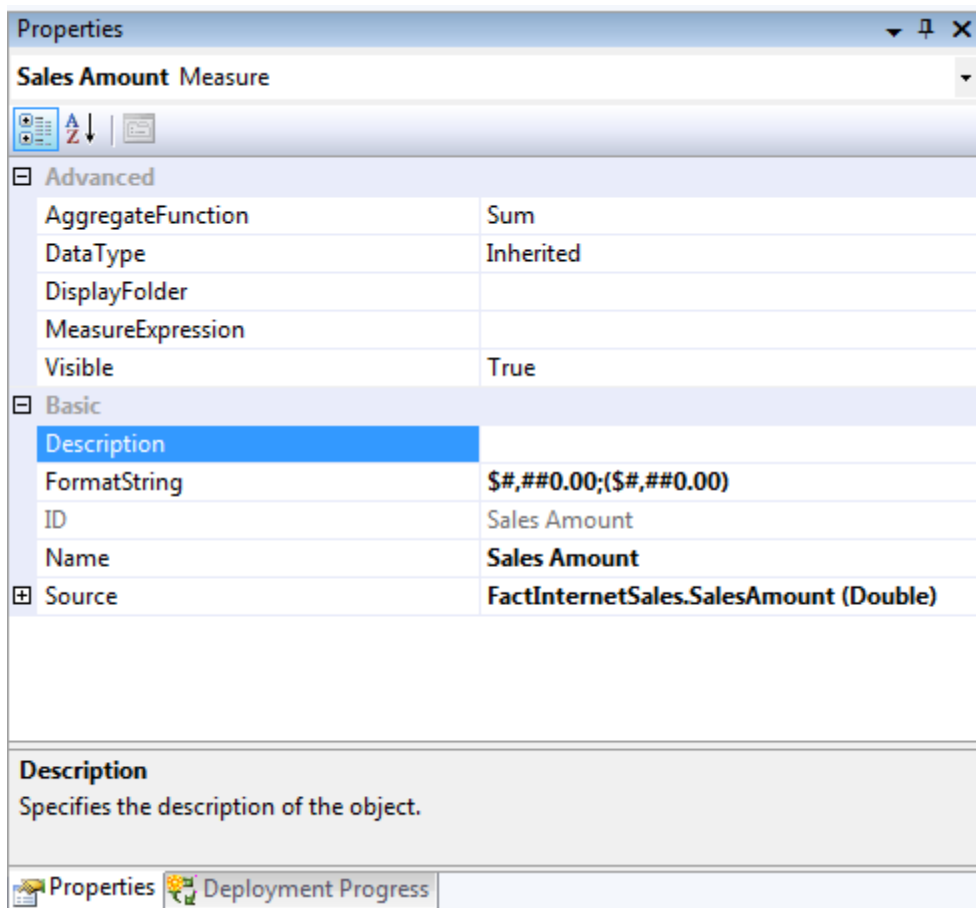


Figure 3–15. You can easily format a measure using the Properties windows in BIDS.

Note As mentioned previously, although it's relatively easy to add derived measures to a cube, we don't often use them in production solutions. We prefer to calculate and store most required measures during the ETL process. If there are measures used by a small number of end users, we prefer to use the resources of SSAS to produce these. This type of measure is called a *calculated measure*, and it is calculated at query time. We'll talk more about calculated measures in Chapter 13, "Introduction to MDX."

Reviewing Dimensions: Attributes

The next step in refining your cube is reviewing the attributes in each dimension that the Cube Wizard created for you. Many times, your end users will not need or want to see all of this information, and because these attributes will be visible to end users, removing unnecessary attributes is a good thing to do. It is a common development practice to remove attributes that are not needed. Refer to the business taxonomy and original documentation you created during the planning phase of your project at this time so that you include only required attributes. Your cube needs to be comprehensive but also usable. Reviewing business requirements will help you get the correct balance. Also, removing unneeded attributes will improve cube processing time and query time. It is also very important to rename attributes to reflect the business language that you captured during the modeling phase. As mentioned previously, your use of business taxonomies dramatically improves cube usability.

To edit the attribute values, names, or properties for a particular dimension, in the lower-left pane (Dimensions) of the Cube Structure tab, click the plus sign next to the dimension you want to edit, and then click the “edit the dimension” link. You will now be working in the dimension editor area of BIDS with the particular dimension you just selected opened in its own editor. The dimension editor surface contains different sections than the cube editor. The dimension editor tabs are Dimension Structure, Attribute Relationships, Translations, and Browser, as shown in Figure 3–16.

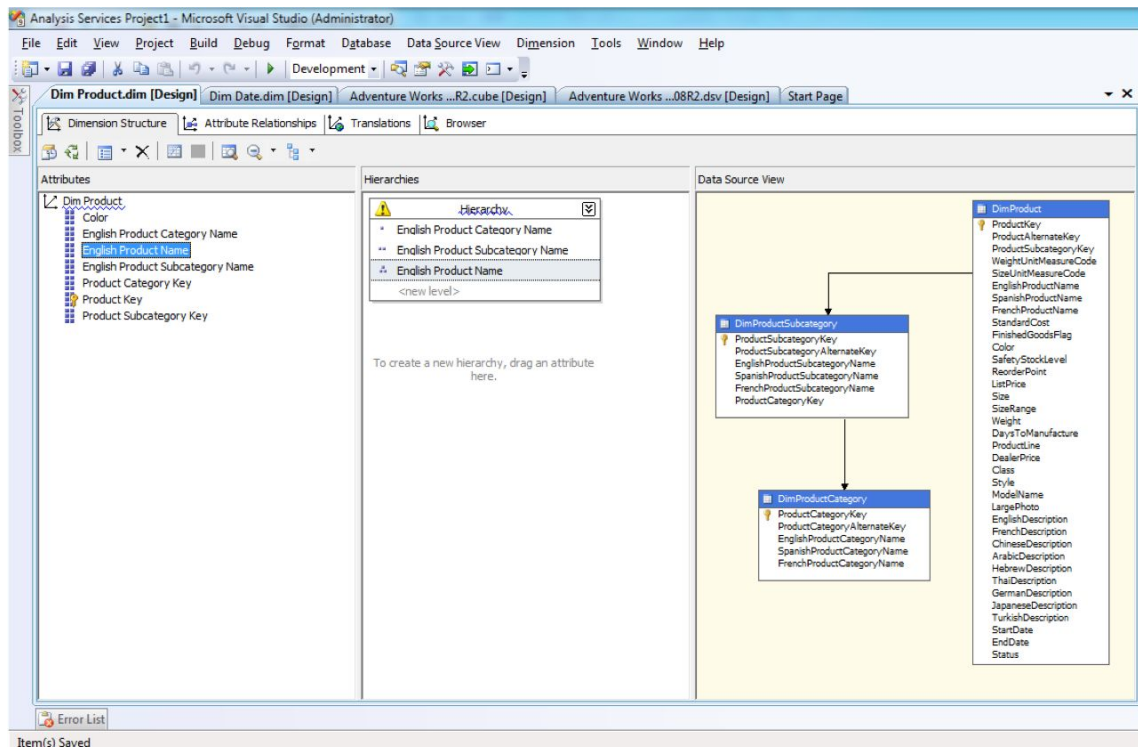


Figure 3–16. You can use the Dimension Structure tab in BIDS to make structural changes to dimension members. These changes can include renaming, deleting, or adding attributes. Changes also include creating or refining dimensional hierarchies.

Tip The default value for each dimensional attribute is the first member loaded into the dimension. A simple change you can make to improve dimension usability is to set the default attribute value that will appear in the end-user client tools for key attributes to a more meaningful value. To do this, with the dimension open in the Dimension Structure tab in BIDS, right-click the attribute, and then click Properties. In the Properties window, click the text box next to the `DefaultMember` property to activate the Build button (gray icon with three dots), and then click on that Build button to open the “Set Default Member - *Attribute Name*” dialog box. In this dialog box (see Figure 3–17), you can select a default value or write an MDX expression.

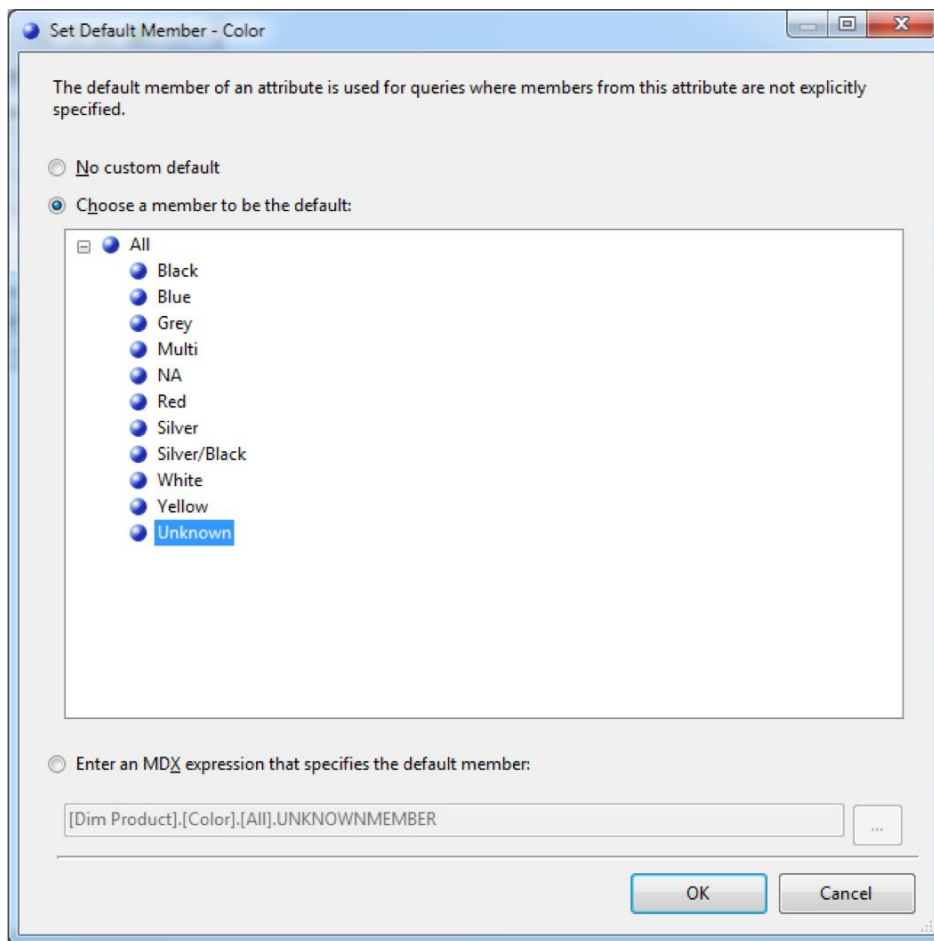


Figure 3–17. You can use the Set Default Member dialog box to change the default value for individual dimension attributes in BIDS. This can improve dimension (and cube) usability.

Reviewing Dimensions: Hierarchies

In Chapter 2, we discussed the concept of a dimensional hierarchy as being a rollup of attributes. By default, BIDS creates only one rollup to an All member. Sometimes during the initial cube build, BIDS will detect additional hierarchies. During the execution of the Cube Wizard, BIDS will present you with a page in the wizard so that you can review (and make simple changes to) any hierarchies that it has detected.

SSAS uses a couple of types of hierarchies. The first type is called a *natural hierarchy* because the source data reflects a one-to-many relationship between the data in one or more columns from the source table. An example of a natural hierarchy is a geography hierarchy, which is built from a Geography source table with a State column and a City column, where each state contains many cities. Usually, when you build your cube in BIDS, SSAS will correctly detect natural hierarchies in the source data and will build hierarchies (using the source column names as level names) in the resulting dimension. You can modify (or create) these types of hierarchies manually as well.

An example of a natural hierarchy that we've had to commonly create in production BI solutions is one or more additional time hierarchies. These are used to reflect different business calendars, such as fiscal, 4-5-4 (retail), manufacturing, and so on.

Another type of hierarchy is a *navigational hierarchy*. As the name indicates, these are groupings for one purpose only—navigation. The source data has no particular relationship between the data in the hierarchy levels. An example using the Customer dimension could be a hierarchy with the State attribute listed above the Gender attribute. You will have to manually create any navigational hierarchies as part of your cube-refinement process. Navigational hierarchies improve cube usability and are easy to add, modify, and delete. All hierarchies associated with a dimension are displayed in the Hierarchies and Levels section of the Dimension Structure tab in BIDS. BIDS does not distinguish between natural hierarchies and navigational hierarchies in the design interface.

To create a new hierarchy or to modify an existing one, select and drag one or more attributes from the Attributes section of the Dimension Structure tab to the Hierarchies and Levels section. You can also rearrange the levels in a hierarchy by dragging the attributes above or below one another in the hierarchies that you've created.

Another design consideration for natural hierarchies only is whether or not you want to configure attribute relationships that reflect the one-to-many nature of the relationship between the data in the hierarchy levels. BIDS allows you to configure this but does *not* set up these relationships automatically. You may want to add this information to natural hierarchies to improve query-processing performance, improve dimension-processing performance, and save disk space. If you create these relationships, SSAS will build internal indexes to make these three tasks more efficient. If you do not create these relationships, then, for example, a query for all customers in the United States would have to touch each fact, rather than traverse a more efficient index.

■ **Caution** If you are going to add these relationships, they must be correct in the data—that is, the data associated must truly have a one-to-many relationship. If it does not, your cube may fail to process, or if it does process, your results may be incorrect. Sometimes BIDS marks incorrectly created attribute relationships as an error by adding a red line under the error data, but BIDS does *not* catch all possible error conditions.

Creating Attribute Relationships

With the release of SQL Server 2008, Microsoft added the Attribute Relationships tab to the Dimension Designer. In this section, you will explore the Attribute Relationship Diagram using the Date dimension. To enhance the Date dimension, and work with attribute relationships, follow these steps:

1. Open the Date dimension in BIDS.
2. Update the Date dimension, by adding or removing attributes, until Calendar Year, Date Key, Day Number Of Year, and Month Number Of Year are the only four attributes remaining.
3. Create a date hierarchy in the hierarchy panel, using Calendar Year, Month Number Of Year, and Day Number Of Year. After completing this step, your Dimension designer should look like Figure 3–18.

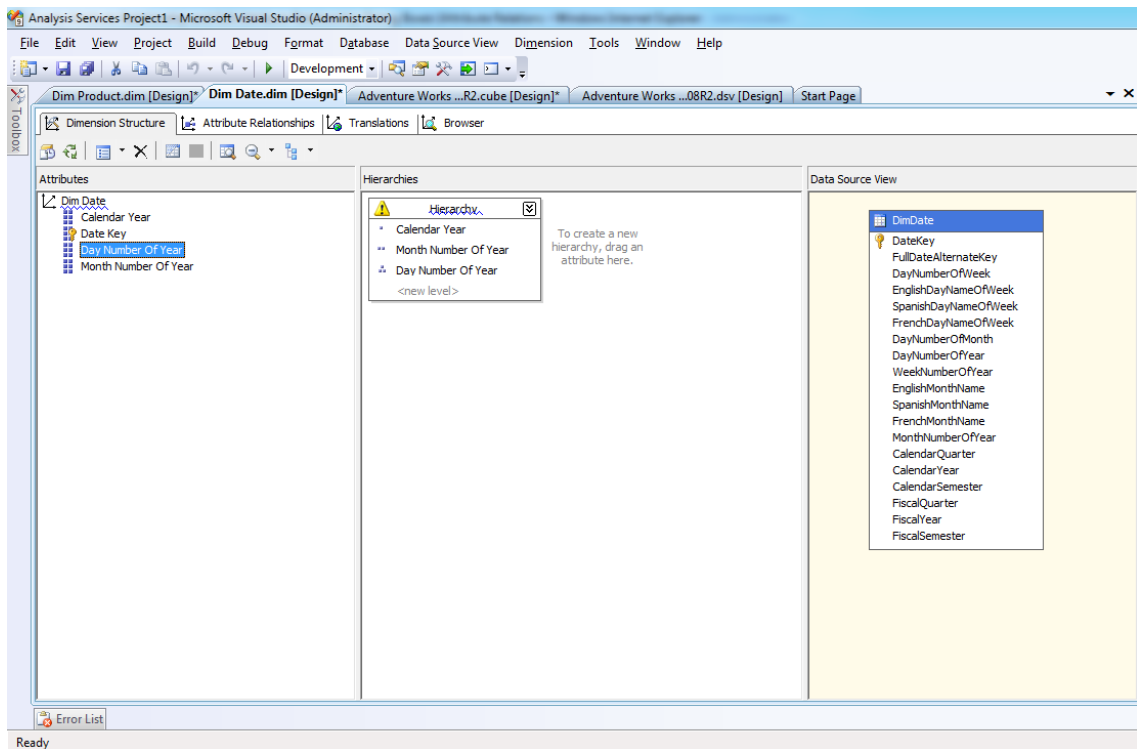


Figure 3–18. In the Date dimension attributes and hierarchy, the blue squiggle under the Hierarchy heading is a warning that attribute relationships are missing.

4. Click the Attribute Relationships tab, which will display the Attribute Relationship Diagram. SSAS takes a first pass in creating our hierarchy and creates a set of default relationships, as shown in Figure 3–19.

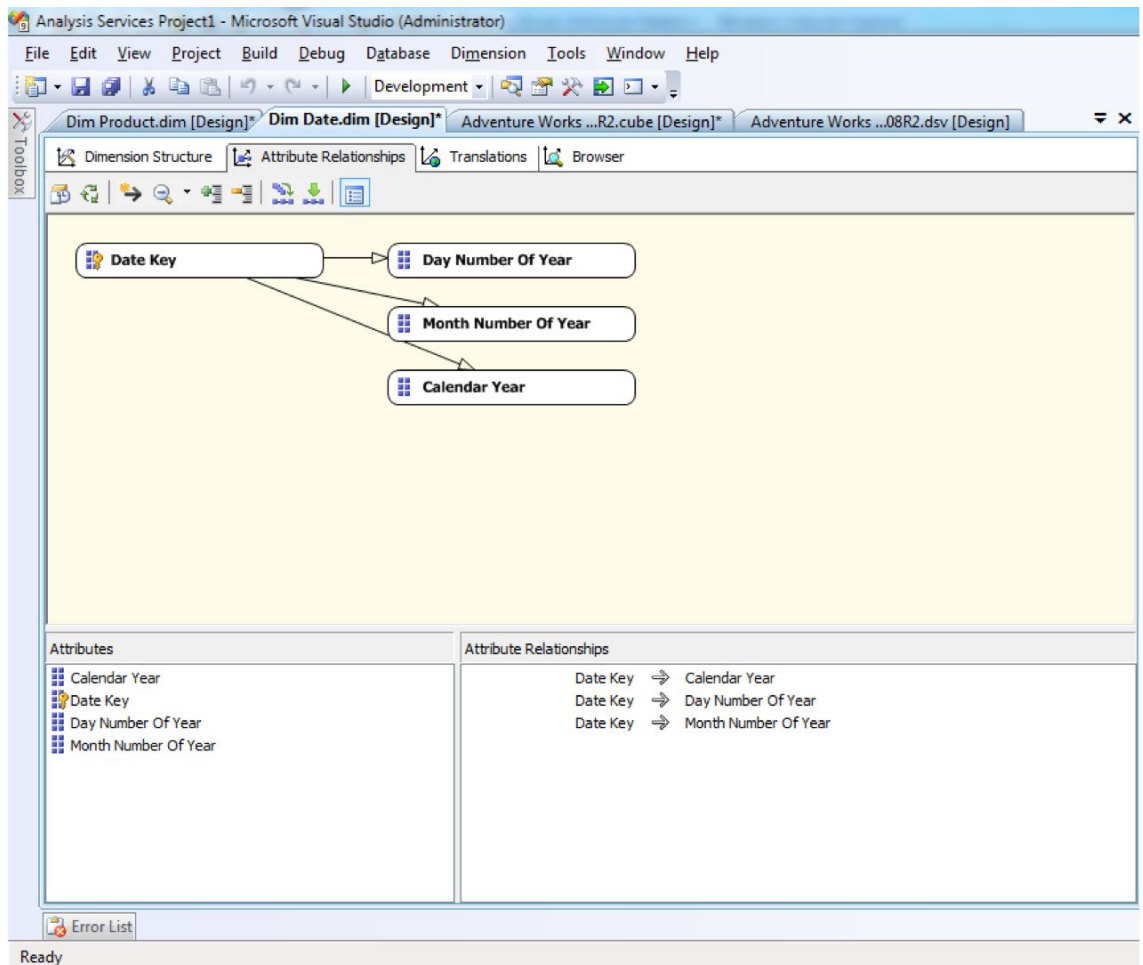


Figure 3–19. The Attribute Relationships tab, with SSAS default relationships displayed.

5. Recalling the previous section, creating these attribute relationships will indicate to the SSAS engine how the hierarchy path actually lies, and which indices SSAS should create. Start by dragging and dropping the Month Number Of Year onto the Calendar Year.
6. Next, drag and drop the Day Number of Year onto the Month Number Of Year. You have now completed defining your parent-child relationships for the Date dimension. Your Attribute Relationship Diagram should now look like Figure 3–20.

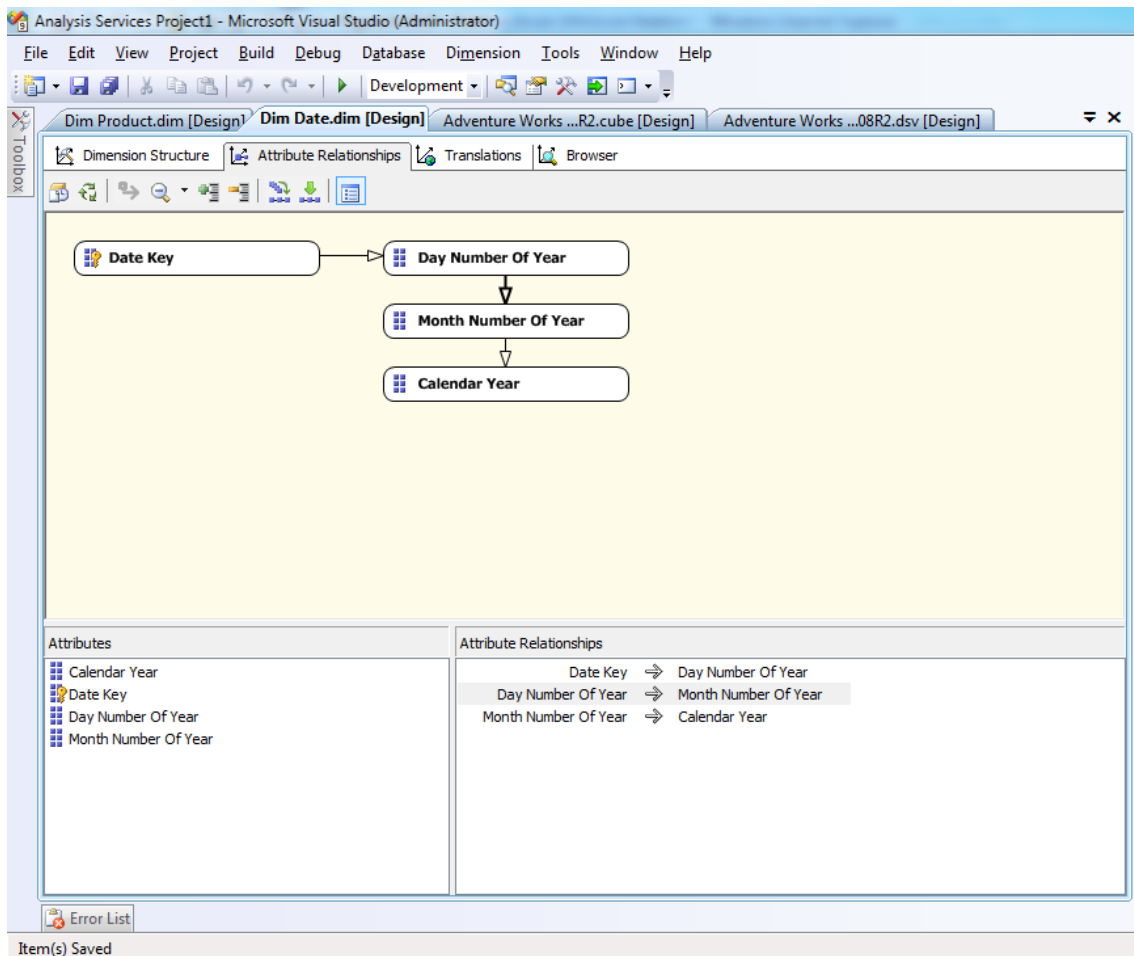


Figure 3–20. The Attribute Relationships tab, with your newly designed relationships displayed. Attribute relationships in hierarchies improve query-processing time and dimension-processing time.

Other Parts of the Dimension Structure Tab

The right pane, Data Source View, allows you to explore the data in the source table or tables from the DSV for this dimension using the same techniques discussed previously (that is, Explore Data, and so on). If you want to modify the table data, for example, by adding a calculated column, then you must edit the DSV in the DSV designer, rather than by using the DSV section of the dimension editor. The DSV section of the Dimension Structure tab is read only.

Similar to the cube browser, the dimension work area also has a Browser tab, so that you can view the dimension, its hierarchies, and its attributes for verification purposes. The other tab available in the dimension work area is Translations, which allows you to supply values for the dimension hierarchy, level, or attribute names that are localized for whatever languages your end users prefer.

Dimension Properties

As with measures, you can also set properties for dimensions, attributes, hierarchies, and levels by clicking the object you want to configure and setting the property using the Properties window in the lower-right corner of the BIDS work area.

An interesting dimension property is `UnknownMember`. This is set for an entire dimension and allows you to specify whether or not unknown member values will be visible in the cube. Also, you can specify a caption, for example, “Unknown” for any unknown members. The Properties window for a dimension is shown in Figure 3–21. One of the reasons this window is so interesting is that SSAS provides you with more flexibility in terms of loading nulls. As you will discover later, you may now configure dimensions to allow nulls and ask that those values be translated to unknowns. This reflects a real-world difficulty of providing perfectly clean data (that is, without null values) to SSAS. We’ll talk more about the configuration in Chapter 6, where we’ll discuss error handling in more detail.

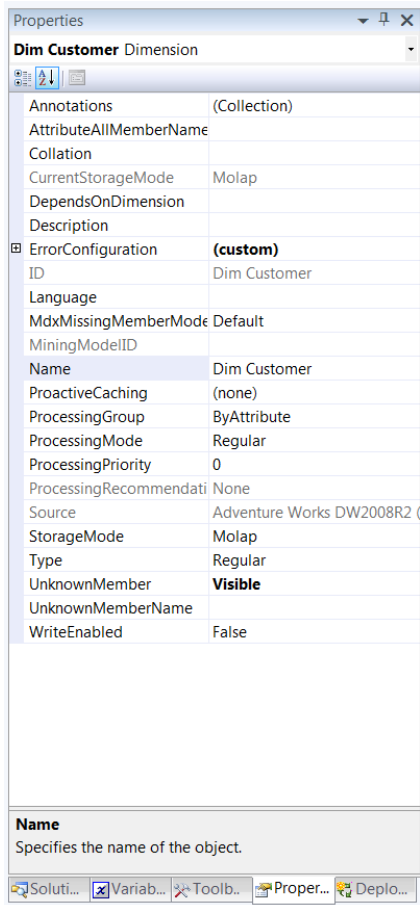


Figure 3–21. The Properties window for a dimension allows you to set advanced properties, including the new `UnknownMember` property. This property is set for the entire dimension only.

It is also common to set properties for attributes or hierarchies using the Properties window. The following are some of the commonly changed settings:

- *Default Member*: This property allows you to set the attribute member value, which will be displayed by default for all end users. The dialog box to configure this was shown earlier. You can also set default members for end users in different security groups, for example, the TexasManagers group can see the Texas state member of a Geography dimension, and so on.
- *Order By*: This property allows you to set the order in which the members will be displayed. The default value is by attribute name.
- *Allow Duplicate Names*: This property allows you to control whether or not duplicate names will be displayed.
- *Member Names Unique*: This property allows you to notify BIDS that all members for a particular level of a particular dimension are unique. The default value of false causes BIDS to issue `SELECT DISTINCT` to load the dimension data. If this setting is set to true, BIDS will simply `SELECT` all members of a dimension at that level. Changing this setting's value to true will reduce dimension-processing times.

Offline vs. Online Mode in BIDS

There are two modes of working with SSAS databases (which contain one or more OLAP cubes) in BIDS: offline and online. When you initially create an SSAS database, you will always be working in offline mode. This means you are creating cube and dimension metadata using the wizards and GUI interface in BIDS. This metadata is saved in file formats proprietary to BIDS and SSAS. The metadata you've created while working offline is not yet deployed to SSAS; you must process the particular structure you are working with to deploy those metadata files to SSAS.

One advantage of working in offline mode in BIDS is that you can make changes to SSAS metadata without affecting end users who are currently connected to your production cube(s). Another advantage is that multiple OLAP designers can work on parts of a solution without affecting end users.

After you've successfully deployed your cube, you may subsequently choose to work offline or online. To work online, you open BIDS, click **File** ► **Open** ► **Analysis Services Database**, click the SSAS server and database name in the **Connect to Database** dialog box, and then click **OK**. If you are working online, any change you make to the metadata is immediately reflected in the production server. There is no **deploy** option available on the project, cube, or dimension menus in **Solution Explorer**. Also, the title bar of BIDS shows the name of the SSAS database and then names of the SSAS server, for example, *Adventure Works DW 2008R2(YourServerName)*, to reflect the fact that you are working online. For the majority of this book, we will use online mode in BIDS.

To work offline after you've successfully deployed your cube, open BIDS, click **File** ► **Open** ► **Project/Solution**, or simply double-click the solution file (*<Projectname>.sln*). The title bar in BIDS shows only the SSAS database name to reflect that you are working offline. In offline mode, after you save changes, you must still deploy the changes to SSAS.

■ **Caution** If you have multiple OLAP developers working offline, you must establish source control procedures outside of BIDS using whatever method works best for you. This could include Visual Source Safe (VSS), Visual Studio Team System (VSTS), SharePoint Portal Server (SPS), and so on. The product you use depends of the type of information you want to place under source control. VSS and VSTS are designed to control code files, and SPS is designed to control specification documents.

The need for external source control when using offline development in BIDS results from the default deployment behavior in BIDS. The default deployment gives you only a single, overly generic message if there are conflicts between the existing metadata on SSAS and new metadata from BIDS. On deployment, BIDS will overwrite anything and everything on SSAS to reflect the entire contents of what is being deployed from BIDS. In other words, BIDS does not produce a list of changes on deployment; it simply overwrites everything.

Other Types of Modeling

SSAS supports many additional capabilities that may affect the final modeling of your cube. These include KPIs, calculations, actions, perspectives, and more. These topics (and more) will be covered in Chapters 4, 5, and 13.

Summary

This chapter covered the processes for creating your first cube. We discussed DSVs in detail, covering their purpose and the options for refining your schema in BIDS. Next, we walked through the Cube Wizard, and then we built our first cube quickly and easily. After that, we refined the measures and dimensions of our cube to make it more efficient and usable.

There's quite a bit more you can do to add features and business value to OLAP cubes. In Chapters 4 and 5, we'll cover the more advanced cube-modeling techniques. In Chapter 6, we'll review the processes involved in physical cube storage.



Intermediate OLAP Modeling with SSAS

After you've mastered basic OLAP modeling and the use of SSAS, the next area to investigate is using some of the intermediate and advanced capabilities of SSAS. This chapter's examples continue to build on the AdventureWorks 2008 R2 sample. The assumption is that you've built a basic cube using the sample database and now want to further enhance your cube. This chapter will cover the following topics:

- Adding key performance indicators (KPIs)
- Reviewing perspectives
- Understanding translations for currency localization
- Covering Actions: regular, drillthrough, and reporting actions

Adding Key Performance Indicators (KPIs)

A KPI is an object that helps cube users to quickly check the status of the most important business metrics. Although you will set up KPIs to return results as numbers, that is, 1 is good, 0 is ok, and -1 is bad, you'll generally display these results as graphics (such as a traffic light graphics with red, yellow, or green selected or different colors or types of arrows). The returned number values aren't as compelling and immediate as the graphics to most end users. SSAS has built-in support to display several types of graphics instead of the numbers. There is also a built-in KPI designer and browser in SSAS. You can use the KPI browser to view the results as the end users will see them (as long as the user's particular client application supports the display of KPI graphics).

Excel and SharePoint Server support the display of SSAS KPIs using the associated graphics defined in SSAS. The KPI view is a tab in the cube designer work area in BIDS (see Figure 4-1).

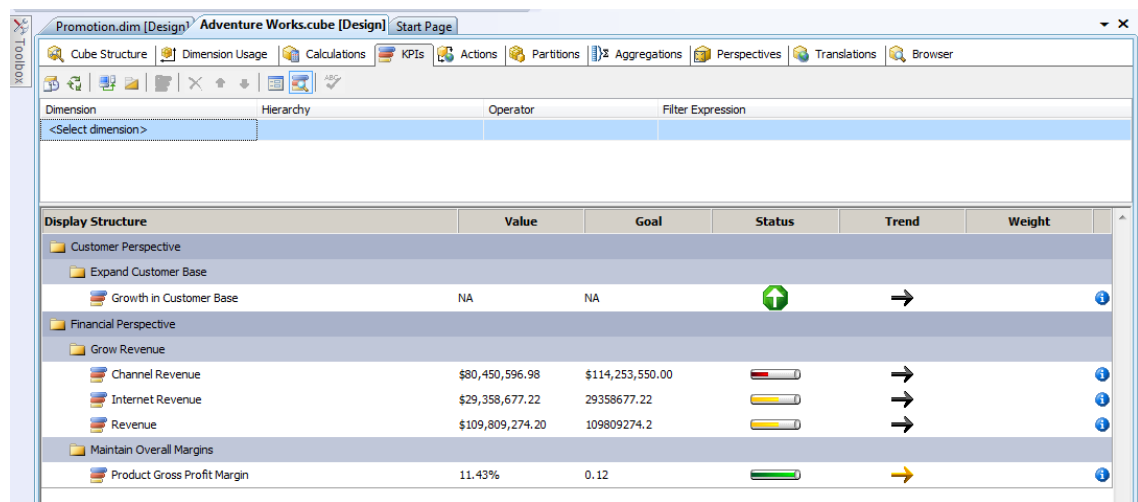


Figure 4–1. SSAS supports the creation of server-based KPIs; BIDS includes a KPI browser.

Implementing KPIs in SSAS

A KPI typically consists of the following four items: Value, Goal, Status, and Trend. KPIs come in two flavors; the three-state KPI, and the five-state KPI. The 3-state KPI communicates the status or trend of your KPI as Good, OK, or Bad. A five-state KPI adds two additional values for rising and falling. KPI definitions are created using MDX, and defined for a particular measure in a measure group. A *measure group* is, as the name would indicate, a grouping of one or more measures in a cube. The AdventureWorks cube includes a variety of measure groups, examples being *Internet Sales* and *Reseller Sales*.

Note Multidimensional Expressions (MDX) is one of the “native” languages used in the SSAS environment (others include XMLA and DMX). MDX is to SSAS as T-SQL is to SQL Server, which means that it is the query language for Microsoft’s implementation of OLAP (cubes). Chapter 13 provides a more complete introduction to the MDX language.

Here are definitions for the four most typically used items in KPIs:

- *Value*: MDX statement that returns the *actual* value of the metric.
- *Goal*: MDX statement that returns the *target* value of the metric.
- *Status*: MDX statement that returns Value – Goal. For a three-state KPI, these are defined as 1 (good), 0 (OK), or –1 (bad). When returning a five-state KPI, these are defined as 1 (good), .50 (rising), 0 (ok), –.50 (risk), or –1 (bad).

- *Trend*: MDX statement that returns Value – Goal over time. For a three-state KPI, these are defined as 1 (good), 0 (OK), or –1 (bad). When returning a 5-state KPI, these are defined as 1 (good), .50 (rising), 0 (OK), –.50 (risk), or –1 (bad).

You can also nest KPIs, that is, create parent and child KPIs. If you do this, you can then assign a Weight value to the nested KPI, which shows the child's percentage of contribution to the parent KPI value. For example, in a growth ratio percentage for a department of a store, nested (or child) KPIs can be used for each department's growth ratio percentage, rolling up to the parent KPI that can reflect growth ratio percentage for an entire store.

The AdventureWorks sample ships with several sample KPIs, one of which is shown in Figure 4–2. The MDX statement to define the Value and Goal properties is straightforward. You can see in Figure 4–2 that SSAS includes some MDX functions specific to KPIs. These functions are used to create the Status value for the example KPI. The MDX functions `KPIValue` and `KPIGoal` are simply aliases for the MDX statements defined earlier for those properties. Note the use of the MDX `IIF` function. `IIF` is analogous to the *Immediate If* construct in several programming languages. `IIF` evaluates the expression passed as the first parameter, and if True, returns the second parameter. Otherwise, `IIF` will return the third parameter. The MDX statement in Figure 4–2 will return 1 (good) if Customer Profitability exceeds its goal and –1 (bad) if it does not.

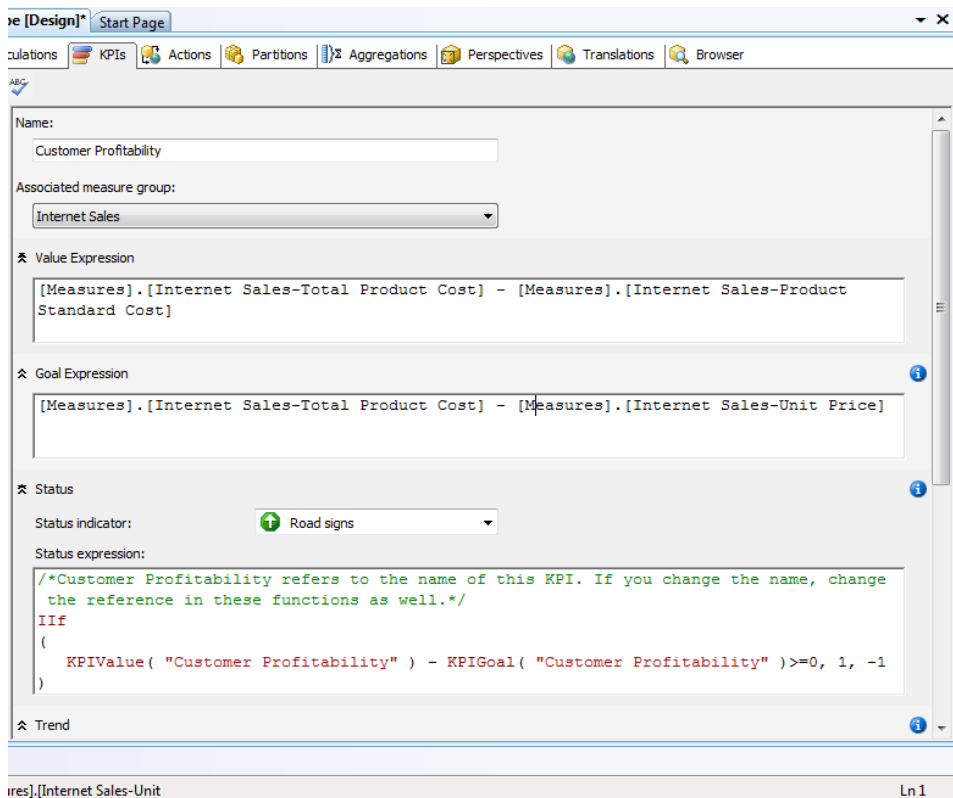


Figure 4–2. The AdventureWorks sample cube includes a KPI called Customer Profitability. Note the use of MDX to define the key properties of the KPI.

KPIs were first introduced with SSAS 2005. Microsoft included a large number of sample KPIs in the AdventureWorks sample cube, which represent a sampling of the most common business scenarios for KPIs. Figure 4–3 shows a partial view of the KPI templates provided.

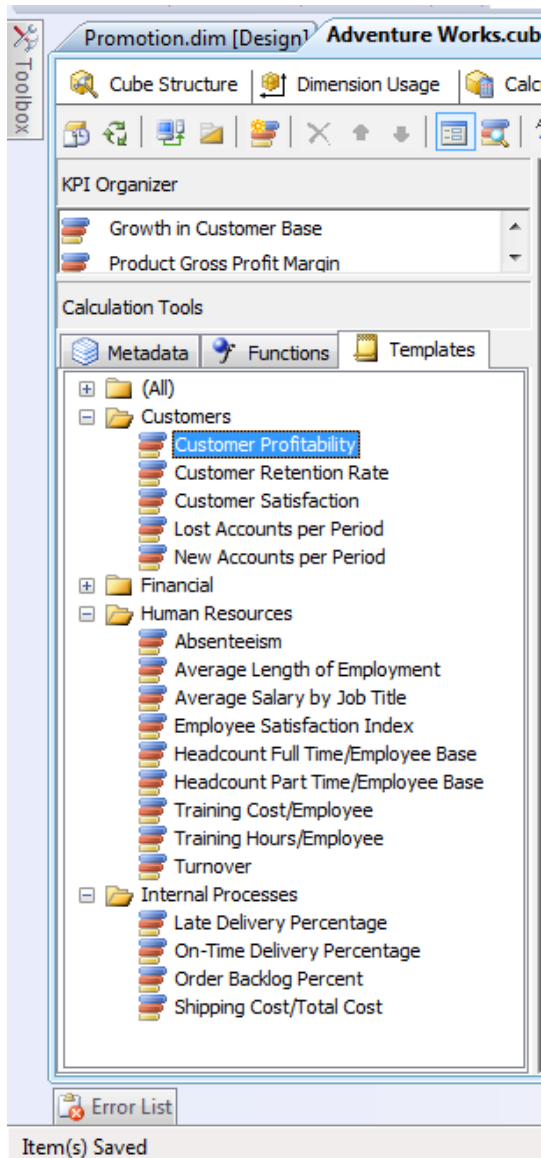


Figure 4–3. Microsoft provides a large number of templates for KPIs in the AdventureWorks sample cube. These samples represent some commonly monitored business metrics.

If you examine these samples, you'll note that they are easy to customize for your particular environment. First, they are generously commented, and second, they are set up as MDX templates. This means they have double chevrons (<< and >>) surrounding replaceable parameters. One section of the templates that warrants a bit more explanation is the Trend expression. Figure 4-4 shows the template for the Customer Retention Rate KPI template as an example of this.

Value, Goal, and Status expressions are self-explanatory. The Trend expression uses the MDX function `ParallelPeriod` to get a value from the same part of a different time period, such as "this fiscal week last year," to support the trend. `ParallelPeriod` is one of hundreds of built-in MDX functions. `CurrentMember`, which returns properties such as Name and Value for the currently selected member, is another built-in function used for the calculation of the Trend value.

■ **Tip** MDX color-codes built-in functions brown in all code windows in SSAS. Also, basic IntelliSense is enabled. So you can type a function followed by an opening parenthesis, and SSAS will display a tooltip with the function arguments.

The `ParallelPeriod` function returns a value from a prior period in the same relative position in the time dimensional hierarchy as the specified value. The three arguments are dimension level (for example, `DimTime.CalendarTime.Years`), a numeric value to say how many periods back the parallel period should be, and a specific value or member (for example, `DimTime.CalendarTime.CurrentMember`).

KPIs Actions Partitions Aggregations Perspectives Translations Browser

Name:
Customer Retention Rate

Associated measure group:
<All>

Value Expression

$$(1 - (\llcorner \text{Number of Customers Lost} \llcorner / \llcorner \text{Total Number of Customers} \llcorner)) * 100$$

Goal Expression

```
/*This can be a fixed value if you know your Customer Retention Rate goal.*/
<<Customer Retention Rate Goal>>
```

Status
Status indicator: Gauge
Status expression:

```
/*Customer Retention Rate refers to the name of this KPI. If you change the name, change the reference
in these functions as well.*/
IIf
(
  KPIValue( "Customer Retention Rate" ) - KPIGoal( "Customer Retention Rate" ) >= 0, 1, -1
)
```

Trend
Trend indicator: Standard arrow
Trend expression:

```
/*The periodicity of this trend comparison is defined by the level at which the ParallelPeriod is
evaluated.*/
IIf
(
  KPIValue( "Customer Retention Rate" ) >
  ( KPIValue( "Customer Retention Rate" ),
    ParallelPeriod
    (
      [ <<Time Dimension Name>> ]. [ <<Time Hierarchy Name>> ]. [ <<Time Year Level Name>> ],
      1,
      [ <<Time Dimension Name>> ]. [ <<Time Hierarchy Name>> ]. CurrentMember
    )
  ), 1, -1
)
```

Additional Properties

Ln 1

Figure 4–4. The KPI template for Customer Retention Rate uses the MDX ParallelPeriod function to calculate the Trend value.

Implementing KPIs in SSMS

New to SQL Server is the ability to create KPIs directly within the SQL Server Management Studio (SSMS). Using the familiar CREATE / DROP syntax, a KPI can now be managed via script in the same way as a table or view. To create a KPI in SSMS, connect to your Analysis Services server and create a new MDX query. Once your new query pane appears, select Adventure Works from the Cube drop-down list. Figure 4–5 shows a blank MDX query, with the Adventure Works cube selected and its corresponding measure groups.

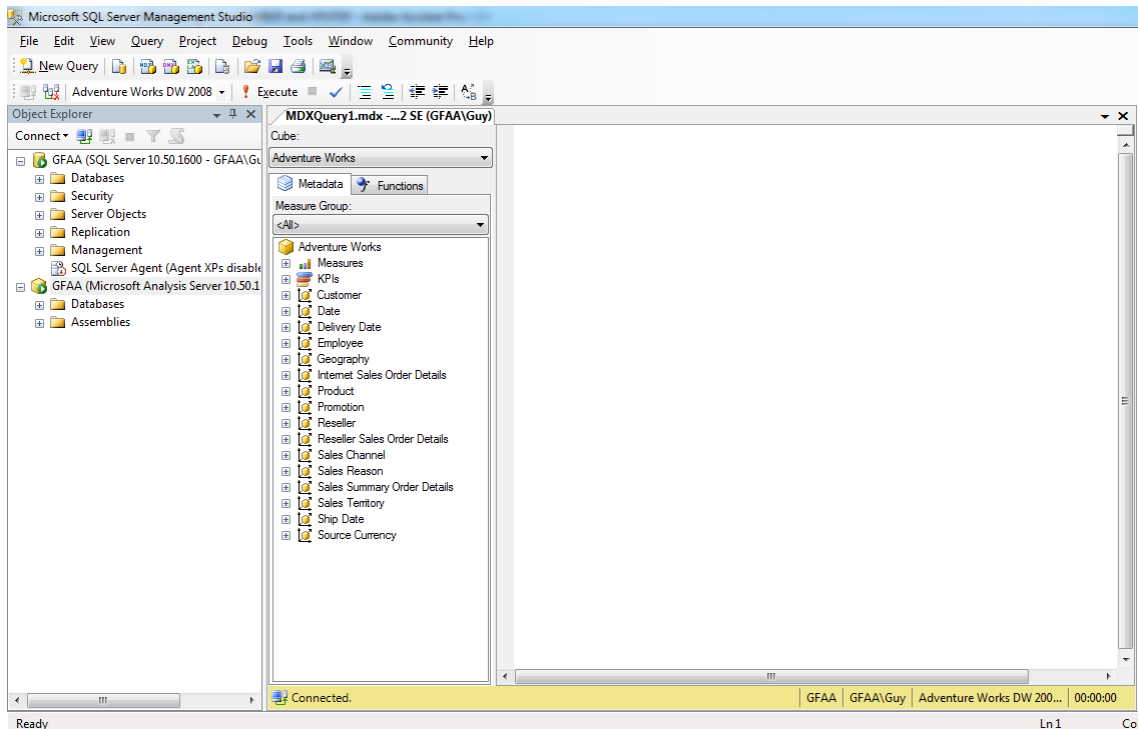


Figure 4–5. A blank MDX query, with selected cube and measure groups displayed.

Creating a KPI with MDX is accomplished by using the Create KPI statement. Your KPI will be defined using Value, Goal, Status, and Trend expressions, just as in the previous section through the GUI. When building your KPIs in SSMS, you can use existing measures or create your own using the Create Member statement. In this section, you will create a new Gross Profit Margin KPI in the Adventure Works cube. To accomplish this, you will also need two new calculated members, Gross Profit Margin Prior Period and Gross Profit Margin Change, to support the Gross Profit Margin KPI.

Create the Gross Profit Margin Prior Period by entering and executing the following MDX:

```
Create Member [Adventure Works].[Measures].[Gross Profit Margin Prior Period]
As
IIf( ([Measures].[Gross Profit Margin], [Date].[Fiscal].PrevMember),
    ([Measures].[Gross Profit Margin], [Date].[Fiscal].PrevMember), 0);
```

Note the use of `PrevMember` in the above MDX. The `PrevMember` function returns the previous member that is at the same level as the member in your expression. In other words, if you are viewing a fiscal month, `PrevMember` will be the prior fiscal month; if you're looking at a fiscal year, `PrevMember` will be the prior fiscal year.

The next calculated member you need to create is `Gross Profit Margin Change`, which will simply calculate the period-over-period change in gross profit margin. Create this member by entering and executing the following MDX:

```
Create Member [Adventure Works].[Measures].[Gross Profit Margin Change]
As
IIf( [Measures].[Gross Profit Margin Prior Period],
    ([Gross Profit Margin] - [Gross Profit Margin Prior Period]) / [Gross Profit Margin Prior
Period], 0);
```

Now that you have your two calculated measures, it's time to create the KPI. The following MDX will create a new KPI named `Gross Profit Margin KPI`.

```
Create KPI [Adventure Works].[Gross Profit Margin KPI]
As
[Measures].[Gross Profit Margin Change],
Goal = .05,
Status =
    Case
        When KPIValue("Gross Profit Margin KPI") >= KPIGoal("Gross Profit Margin KPI") Then 1
        When KPIValue("Gross Profit Margin KPI") > 0 And
            KPIValue("Gross Profit Margin KPI") < KPIGoal("Gross Profit Margin KPI") Then 0
        Else - 1
    End,
Status_Graphic = 'Gauge',
Caption = 'Gross Profit Margin Change',
Display_Folder = 'KPIs';
```

The `Gross Profit Margin KPI` creates a performance goal for gross profit margin of 5 percent. The status of this KPI is good when the goal is exceeded, and OK when gross profit margin rises but does not exceed the 5 percent goal. Finally, the status is flagged a bad when gross profit margin falls. Note the use of the `Display_Folder` property; this gives you the ability to organize your KPIs. When you are finished, you can use the following `Drop Member` and `Drop KPI` statements to remove these items from the Adventure Works cube.

```
Drop KPI [Adventure Works].[Gross Profit Margin KPI];
Drop Member [Adventure Works].[Gross Profit Margin Prior Period];
Drop Member [Adventure Works].[Gross Profit Margin Change];
```

Using Perspectives and Translations

Perspectives and translations are two features that are supported only in the Enterprise Edition of SSAS. They allow you to create custom views of a cube. Perspectives allow you to create view-like subsets, and translations allow you to associate alternate metadata labels for cube, dimension, and other object names (such as KPIs).

Perspectives

A *perspective* is a named MDX query that appears to the end user as if it were a separate cube. Much like a view in a relational database, perspectives are simply subsets of the features of a cube. When deciding whether to use perspectives, consider the following:

- Perspectives are only supported in the Enterprise and Developer Editions of SSAS.
- If you chose to use perspectives, then your client applications must support them.
- Perspectives are best used when the base cube has a complex structure and simplification would improve usability.

Perspectives are simple to create using BIDS. While working in the cube designer area, you will see a tab devoted to Perspectives. To create a new Perspective, right-click anywhere on that design surface and then click *New Perspective*. Figure 4–6 shows that you can choose to include, or not include, the following items from a cube:

- Measure groups or individual measures
- Entire dimensions or individual dimension members (hierarchies and/or attributes)
- KPIs
- Custom MDX calculations defined on areas of the cube, including calculated members, named sets, and MDX scripts

The built-in cube browser in BIDS supports browsing cube perspectives, so you can verify that the perspective you've created matches the business needs of your particular scenario. Although an MDX query is being generated in BIDS when you create a perspective, there is no way for you to directly edit the MDX statement when using BIDS.

■ Note Unlike relational views, a cube perspective is not a security mechanism. All security permissions defined on the base cube are enforced when any end user browses the cube via a perspective. As stated previously, the primary purpose of a perspective is to provide a simplified view of the base cube to different types of end users.

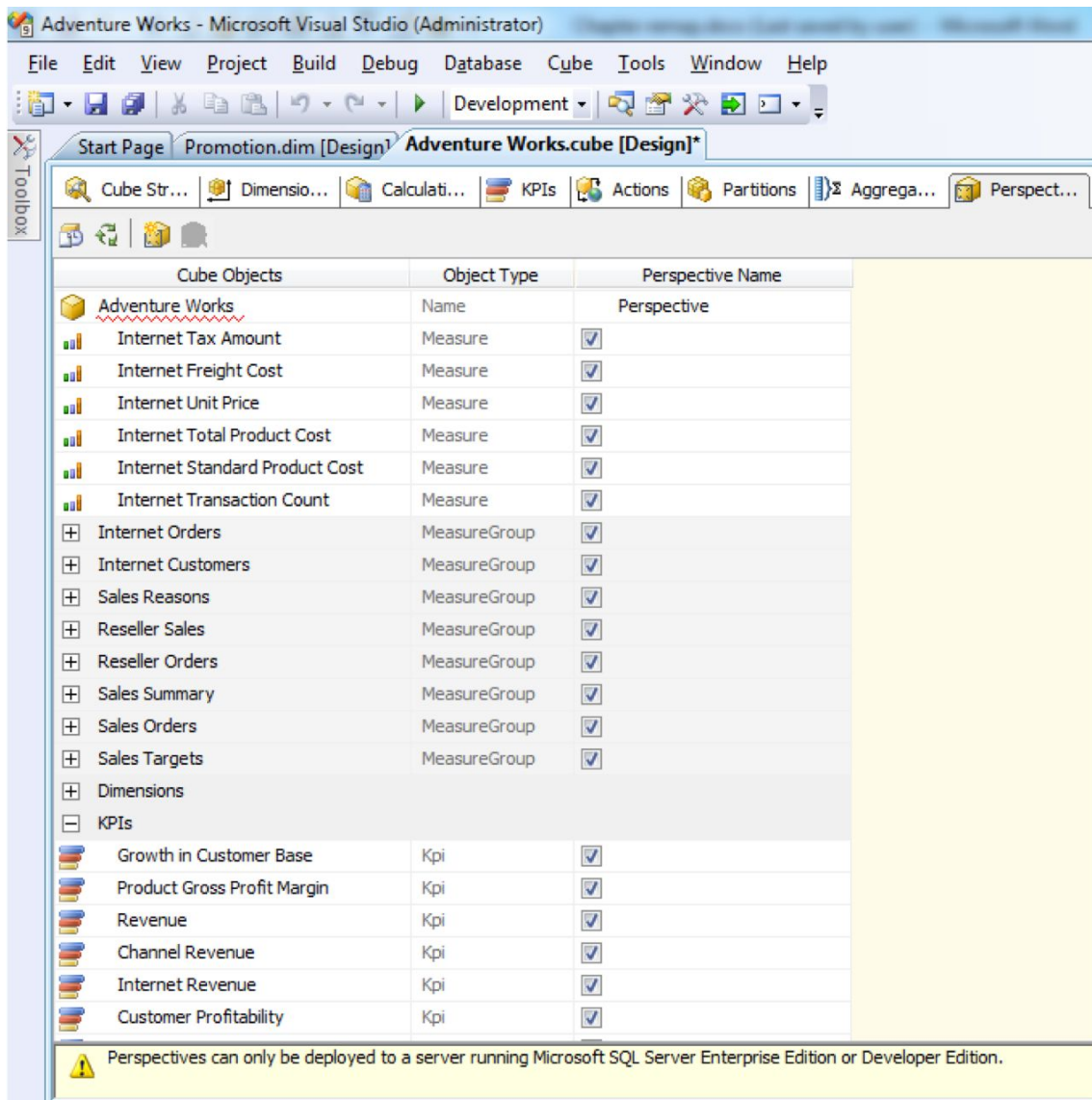


Figure 4–6. SSAS perspectives allow you to quickly and easily create simplified views of your base cube.

Translations

A *translation* is a view of cube information in a different language. It's a view of cube metadata (which you can also think of as the row, column, and filter labels of a pivot table client for an OLAP cube) in the language that you select when defining the Translation.

As with perspectives, translations are simple to create in BIDS. While in the cube work area, you click the Translations tab, and right-click anywhere on the design surface to add a new translation. After the translation has been created, client applications that connect to the cube will display the cube metadata in the language that has been defined in the locale settings in the operating system's Control Panel. Translations, like perspectives, require the Enterprise or Developer Edition of Analysis Services.

Note If the cube does not contain a translation for an end user's particular locale, the default language/translation view is presented to that user.

Figure 4–7 shows a Spanish translation for the sample cube. Note that in this view, you can only add translations for a dimension name, not for dimensional attributes, hierarchies, and levels.

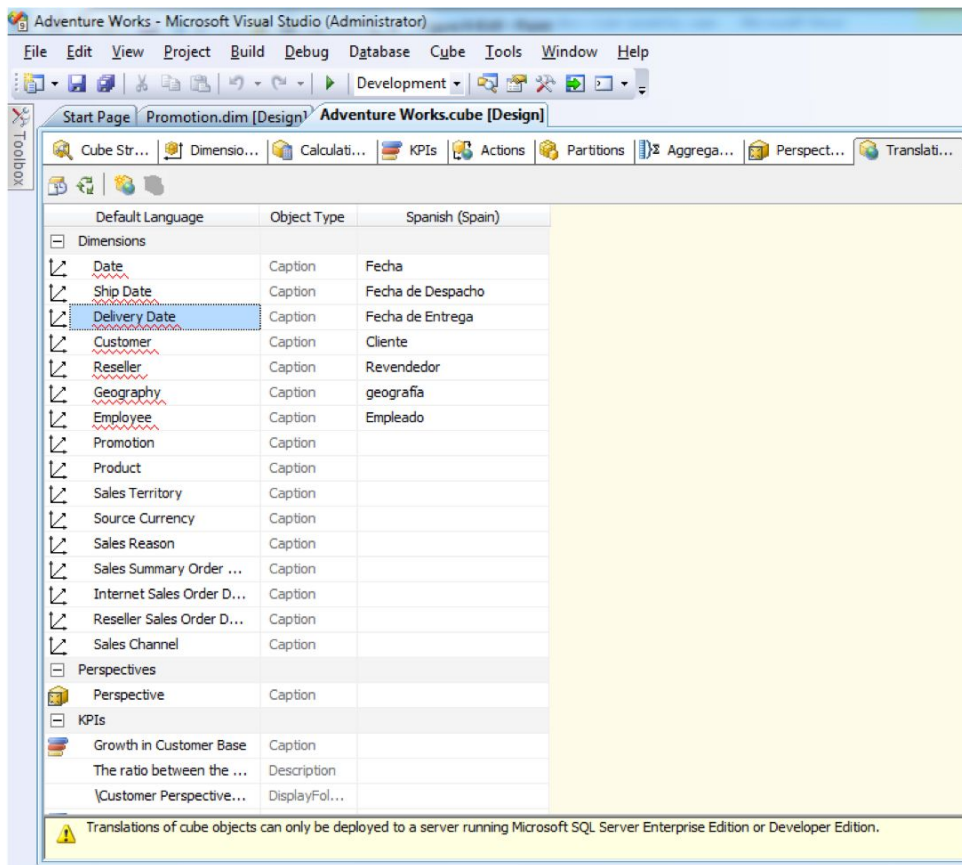


Figure 4–7. Creating translations of cube metadata in BIDS allows you to easily localize the metadata for your cube.

The only complexity with defining translations is that you define only cube metadata (that is, measure names, dimension names, KPI names, and so on) translations using the preceding process. This can include defining the localized name of the dimensions.

To define localized names for dimensional attribute values (for example, in the Sales Reasons dimension, some of the attribute values are manufacturer, on promotion, and so on), you first open the dimension and then add a translation for that dimensional hierarchy, level, or attribute (or other value, such as for the All member and the Unknown member) in the dimension work area. The BIDS cube and dimension browsers allow you to view your cube and dimension translations. You do this by selecting the particular language from the drop-down list at the top of the browser page.

Localizing Measure Values

Another consideration when localizing a cube is to express the value of measures (usually currency or dates) as localized values. If your business scenario requires this type of localization, you will most generally create custom MDX queries to translate and format these values for a particular user. To localize date values, you still must do this manually; however, SSAS offers the *Add Business Intelligence* feature for currency localization. This feature is accessed via a wizard in BIDS to make localization via MDX scripts much simpler.

Note Before you run the Add Business Intelligence Wizard to generate an MDX script for implementing currency conversion in BIDS, you must set up three required prerequisites: at least one currency dimension, at least one time dimension, and at least one rate measure group. For more specifics on the required structures of those dimensions, see the SQL BOL topic, “Currency Conversions (Analysis Services - Multidimensional Data).” The Adventure Works sample meets these three prerequisites.

To access the Add Business Intelligence Wizard, you click the Cube menu in BIDS. Clicking Next on Add Business Intelligence launches the first page of the wizard (see Figure 4–8), which lists possible enhancements for the cube. The choices offered vary depending on whether you are working in the cube design or a dimension design window.

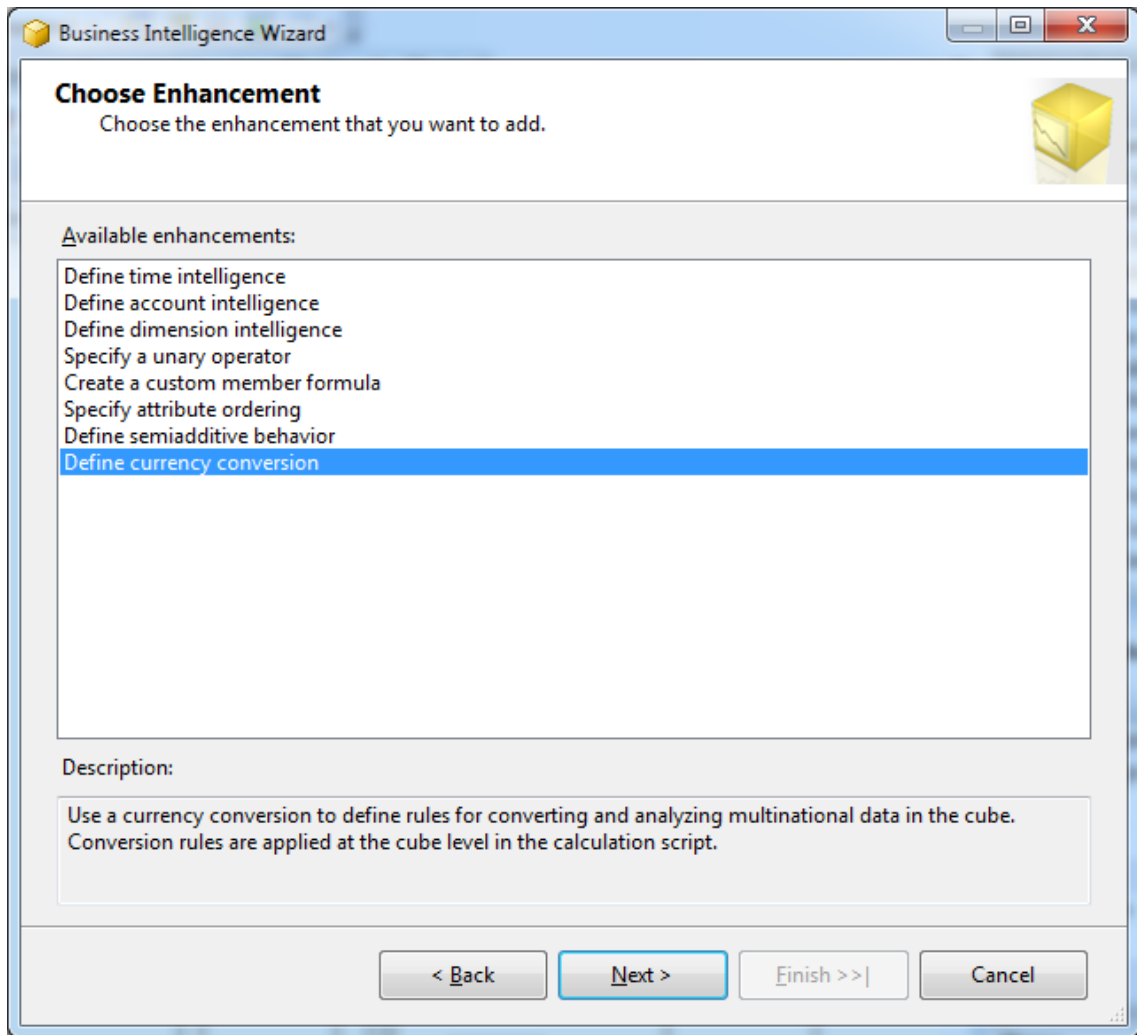


Figure 4–8. The Business Intelligence Wizard in BIDS allows you to quickly and easily localize cube currency values by taking you through several steps of a wizard. This will generate an MDX script and make structural changes to your cube to support currency localization.

In the Set Currency Conversion options page of the wizard (see Figure 4–9), you are asked to make three choices. First, you'll have to select the measure group that contains the exchange rates, then the pivot (or translating) currency, and finally the method you've used to define the exchange rate. You'll define later in the wizard exactly how the translation between source currency (or currencies) and destination currency (or currencies) will be performed.

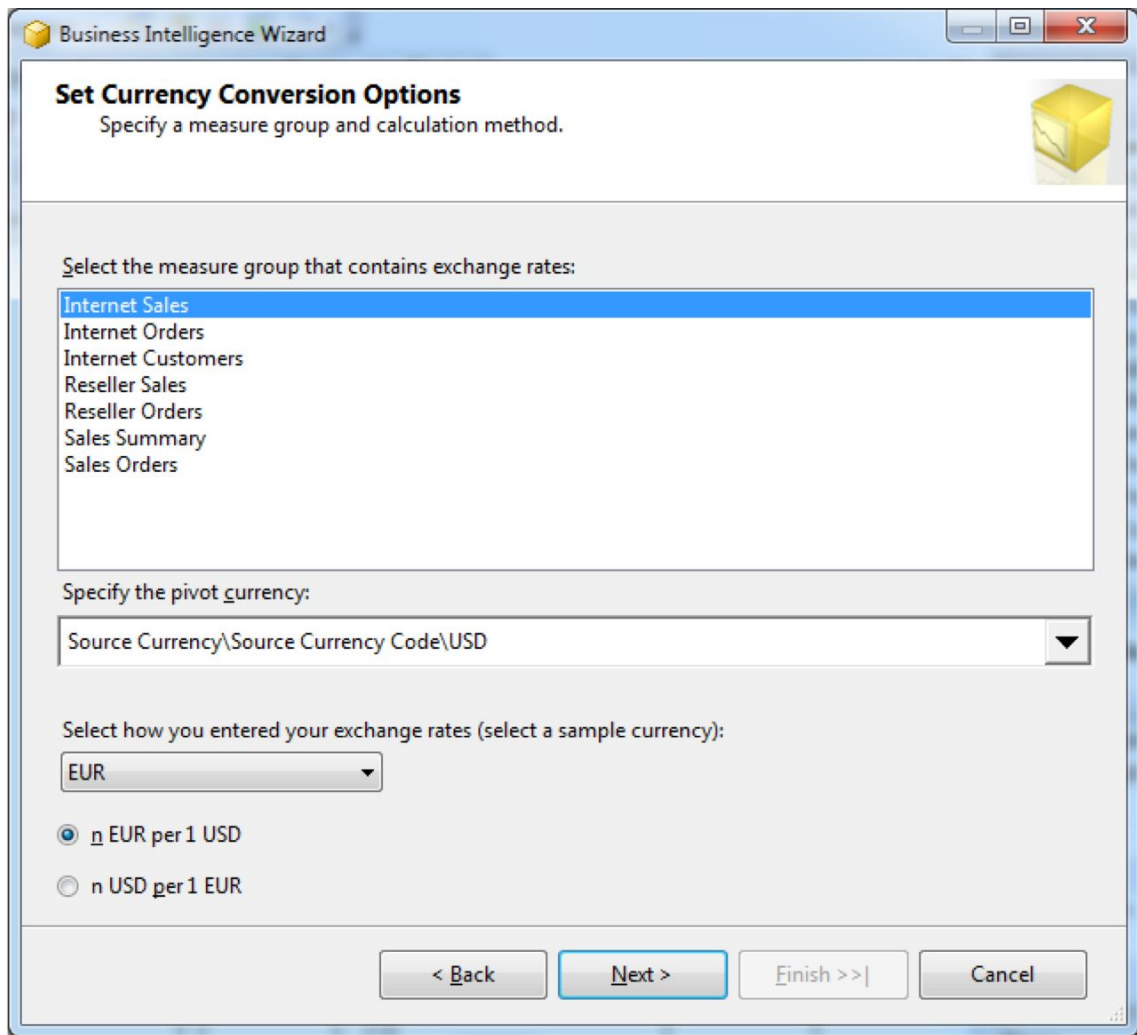


Figure 4–9. Using the Business Intelligence Wizard allows you to localize currency values in your cube.

In the Select Members page of the wizard, select the Sales Amount measure, which will also populate the Exchange Rate Measures column with Internet Sales Amount. Your Select Members page will now resemble Figure 4–10.

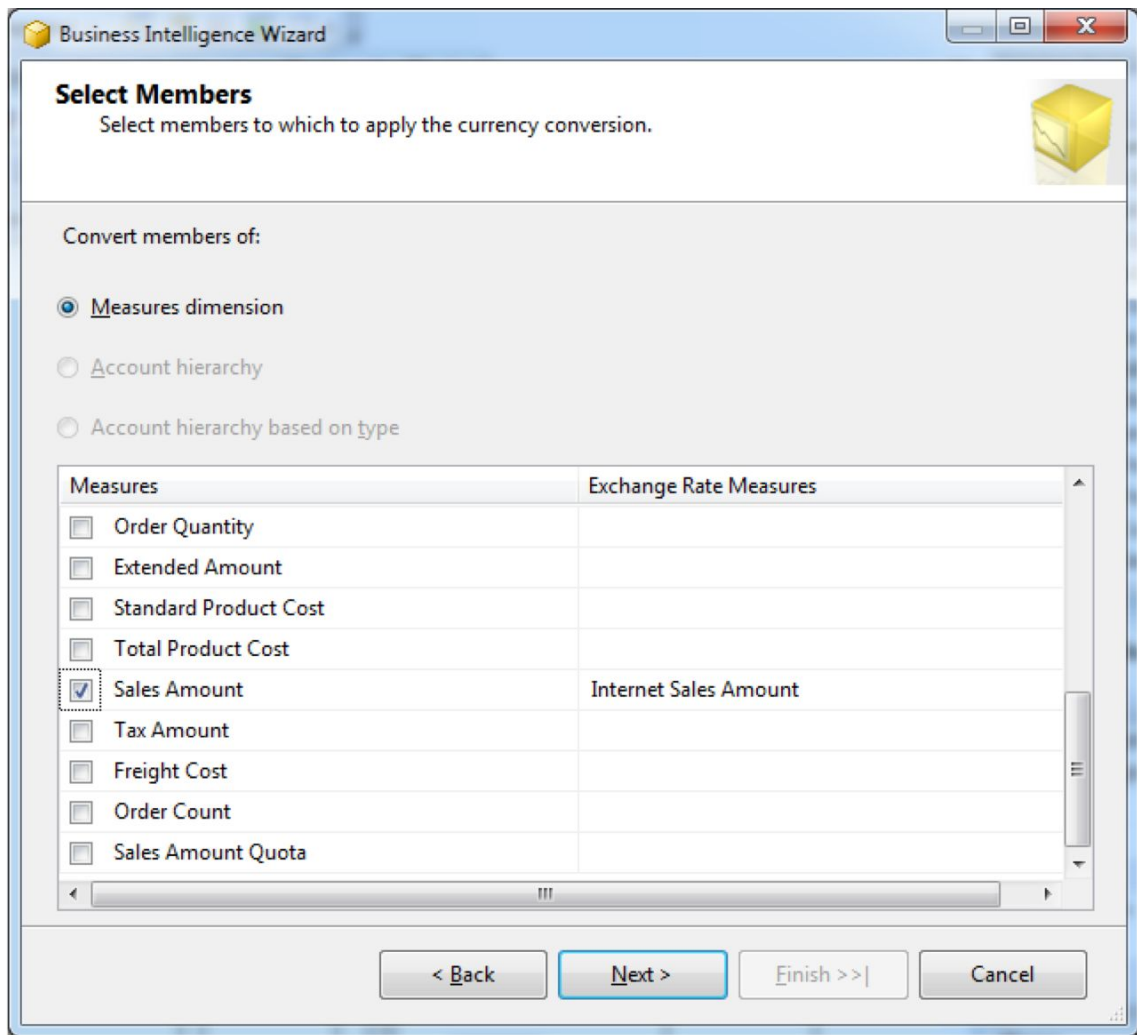


Figure 4–10. In the *Select Members* page, you define which measure values you want to convert to localized currency values.

Click Next, and you are in the most interesting part of this wizard. Here, you are presented with three options for performing the currency localization by selecting a cardinality option: many-to-many, many-to-one, or one-to-many. Although there is a description of each option on the wizard page, the options are summarized here:

- *Many-to-many*: The source currencies are stored as local currencies, translated to the pivot currency, and then translated into one or more reporting currencies at reporting time. For example, you could translate currency EUR, JPY, and GBP, using the pivot currency, USD, into multiple destination currencies CNY, MXN, INR.
- *Many-to-one*: The source currencies are stored as local currencies and then translated to the pivot currency at reporting time. All sources use the pivot currency as the reporting currency. For example, you could translate currency EUR, JPY, and GBP all into USD.
- *One-to-many*: The source currency is stored as the pivot currency and then translated into one or more reporting currencies. For example, you could store values as USD in your fact table and then translate those USD values into multiple destination currencies.

Figure 4–11 shows the Select Conversion Type Wizard page where you make this selection.

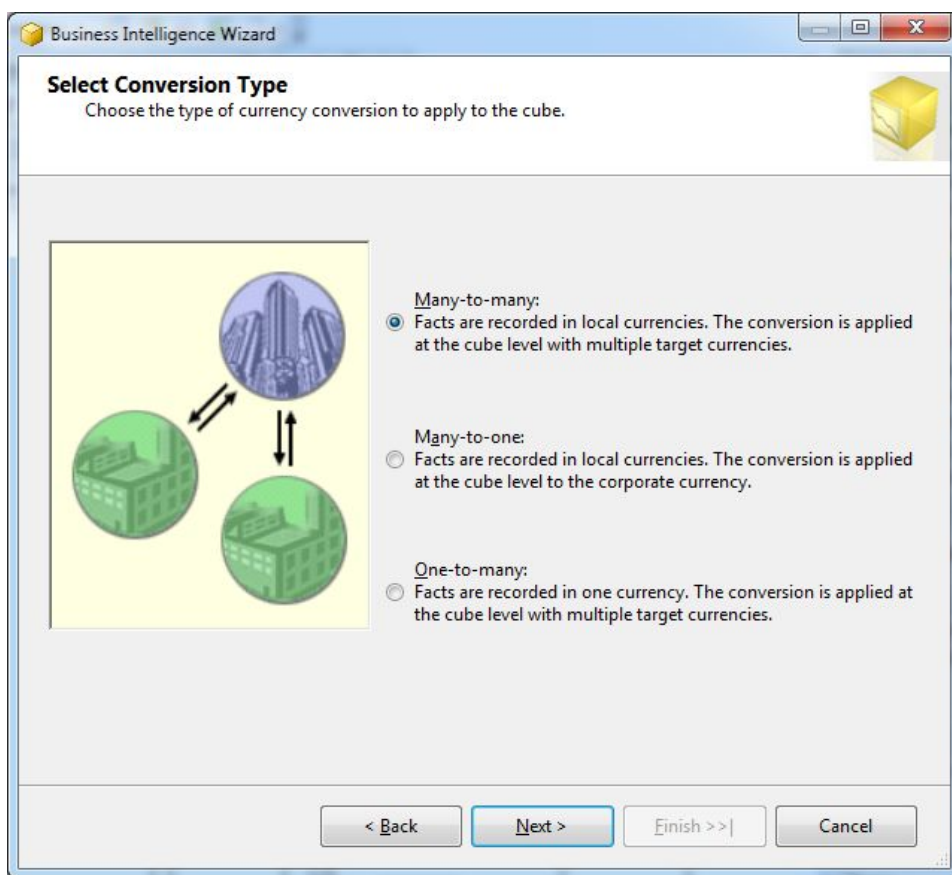


Figure 4–11. An important step in the Business Intelligence Wizard is selecting the cardinality option.

After you select the “Many-to-many” cardinality, you have two more choices in this wizard. You must identify whether a column in the fact table or an attribute value in a dimension should be used to identify local currency values. Choose “Identifiers in the fact table”, followed by Source Currency Code from the drop-down list. Click Next. The last step is to select your reporting (or destination) currencies; choose the Euro (EUR) and the British Pound (GBP). Click Next again. The wizard now has enough information to generate a MDX script to execute your currency calculation. You can see this script in the final step of the wizard (see Figure 4–12).

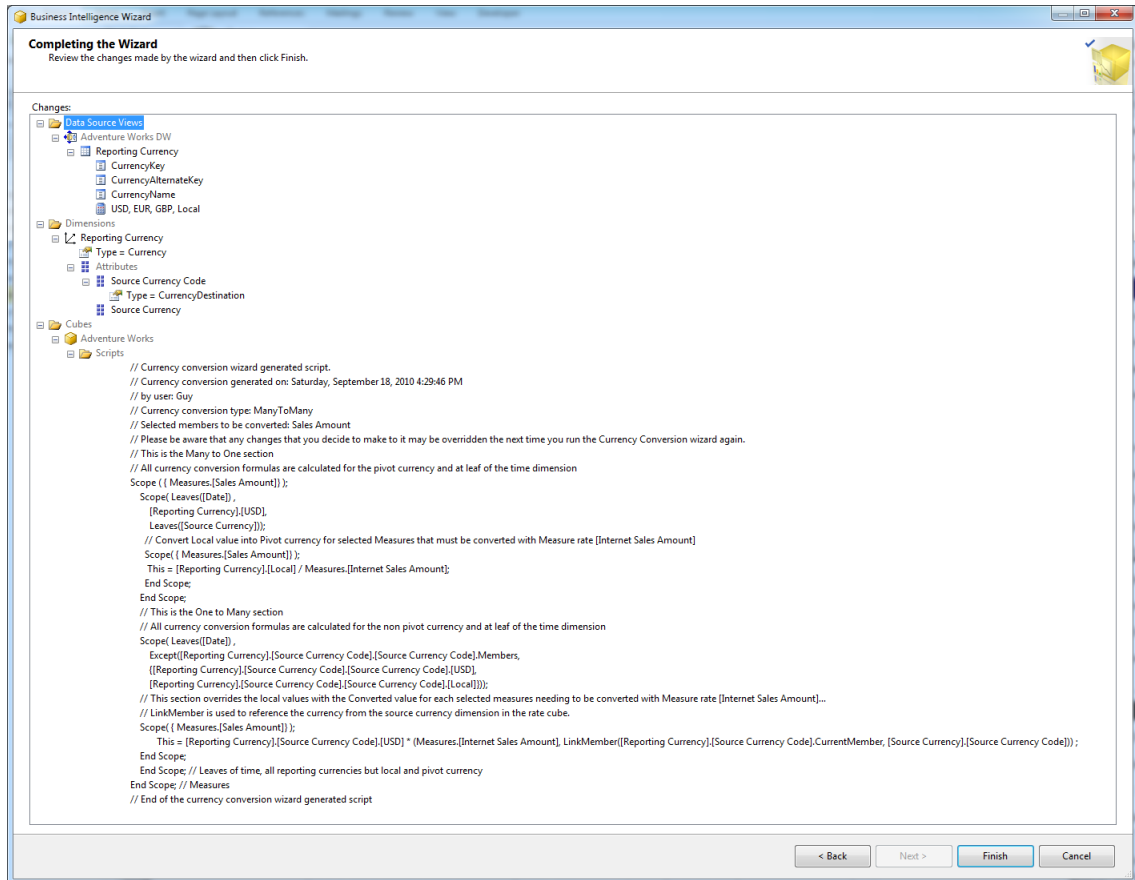


Figure 4–12. The BIDS Business Intelligence Wizard allows you to generate a sophisticated MDX script by answering a series of questions in the wizard pages.

The Business Intelligence Wizard is a powerful tool. As we continue to explore advanced cube modeling in Chapter 5, we will review other capabilities built into the wizard.

Using Actions

An *action* is an activity that an end user can perform by right-clicking either cube metadata (a row or column label) or cube data (a measure value). Actions are added to SSAS cubes and targeted at a particular section, that is, an entire dimension, a particular hierarchy, a particular measure group, and so on. The ability to use cube actions is completely dependent on the type of client application your BI solution uses. For example, Excel 2007/2010 supports several actions, including URL, reporting, and drillthrough actions.

Creating Actions in SSAS

Actions are created in BIDS using the cube designer work area (Actions tab) by completing the property values and writing MDX scripts. As with many other advanced cube-design techniques, the samples that ship with SSAS include several examples of actions.

After you've verified that your particular client applications support SSAS cube actions, you can select one to add from the following action types:

- *(Regular) Action:* Allows end users to right-click either cube metadata or cube data and to perform an activity by passing the value of the cell clicked to one of the following action types: DataSet, Proprietary, Rowset, Statement, or URL. What is returned to the end user depends on which action type has been set up in BIDS. If the action is a URL action, a web page is returned. Rowset actions return rowsets to the client, DataSet actions return datasets, and statement actions allow the execution of an OLE DB statement. Figure 4–13 shows the syntax for a URL action.

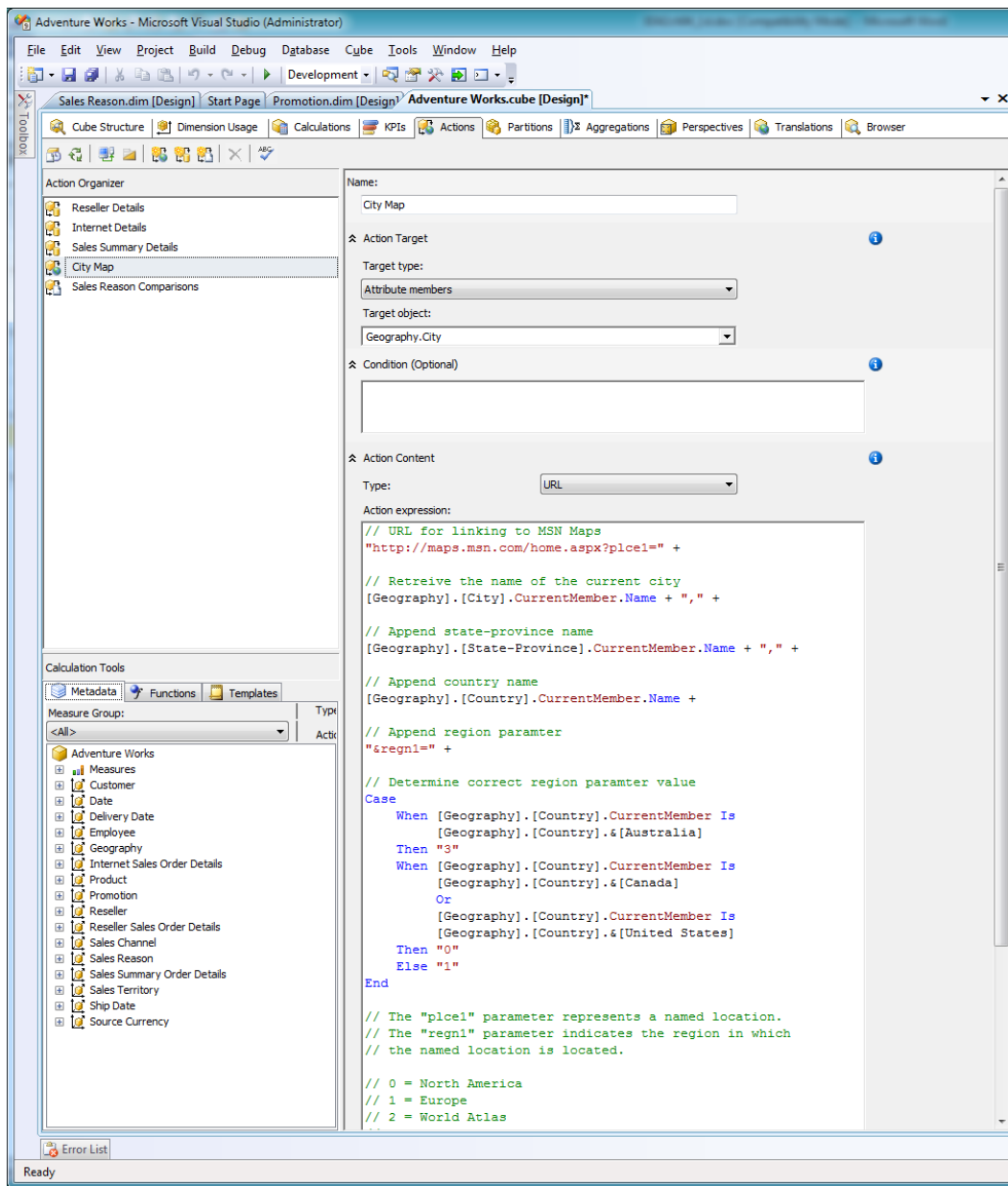


Figure 4-13. URL Actions allow end users to launch Web pages by right-clicking cube metadata or data in their end-user tool environment.

Note For (regular) actions and reporting actions, you must determine the value of the target type and target object properties. As their names suggest, these properties determine where in the cube these custom actions will be available to end users. In Figure 4–13, end users must be viewing the [Geography].[City] attribute to be able to use the CityMap URL Action.

- *Drillthrough Action:* Allows end users to see detailed information “behind” the value of a measure; that is, for this discount amount, what are the values of the underlying dimensional attributes? This is shown in Figure 4–14 using the cube browser. Unlike (regular) actions and reporting actions, creating drillthrough actions requires that you specify a target measure group. Figure 4–15 shows an example of the syntax used to create a Drillthrough Action targeted at the Reseller Sales measure group.

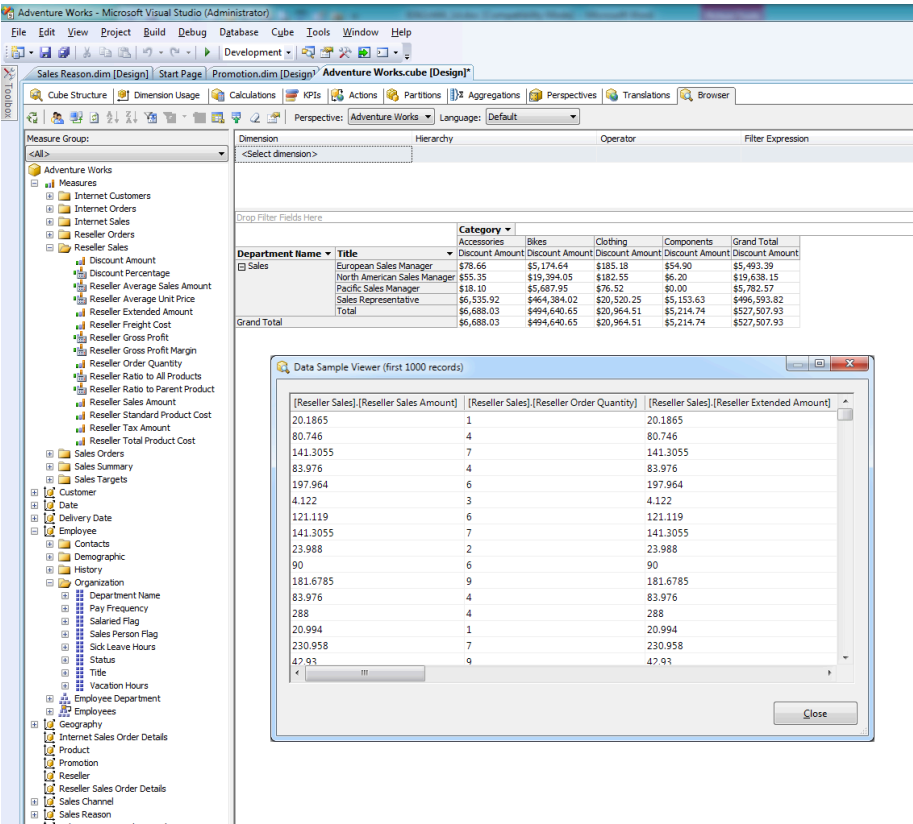


Figure 4–14. Drillthrough Actions allow end users to see the detail behind the measure value for a particular measure.

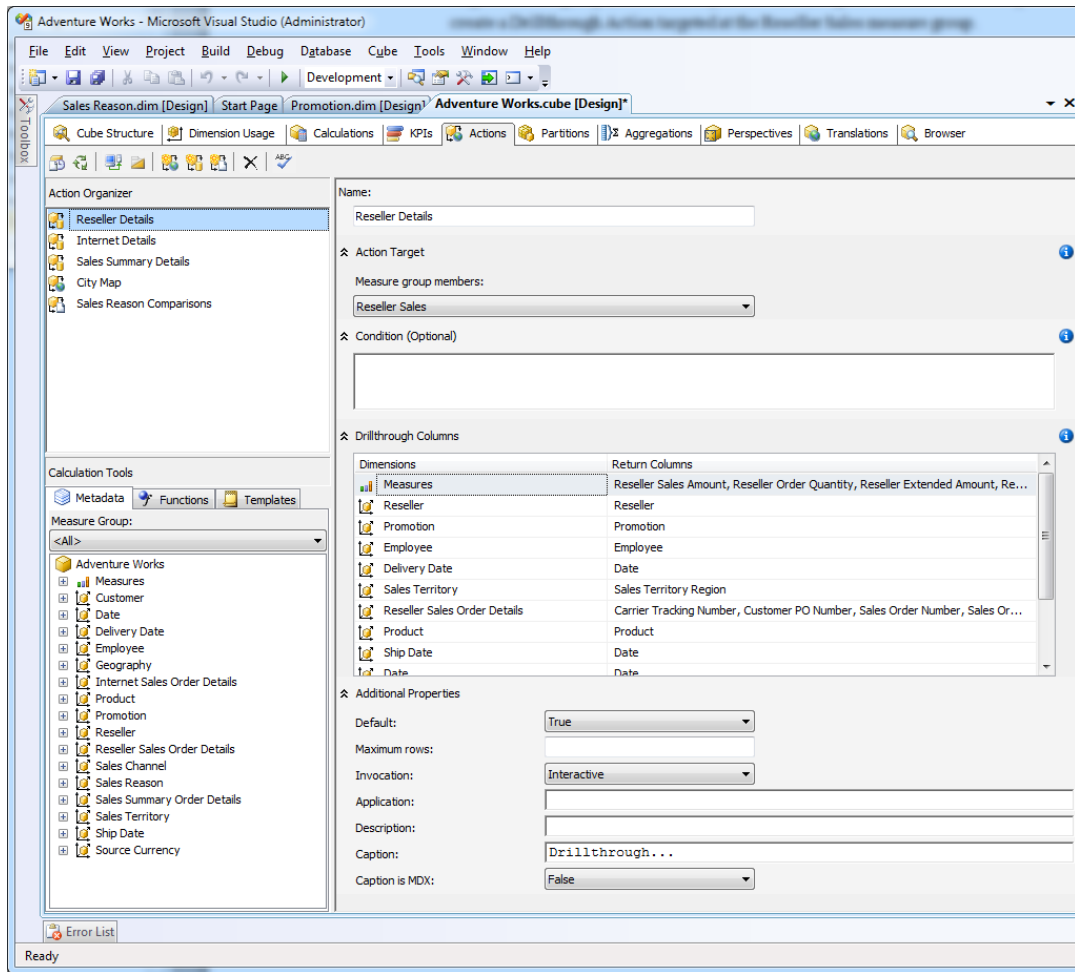


Figure 4–15. Drillthrough actions are targeted at particular measure groups rather than attributes or hierarchies.

- Reporting Action:** Allows end users to right-click a value and will pass the value of the location clicked as a parameter to SSRS. Activating the action causes SSRS to launch using the custom URL, which includes the cell value and other properties (for example, the format of the report: HTML, Excel, or PDF). Figure 4–16 shows an example of the syntax used to create a reporting action. In the Invocation drop-down list in the Additional Properties section, you can specify whether the defined action should be started by the end user (the Interactive option) by right-clicking the client interface pivot table cell, or should be implemented automatically (the On Open option) when the end user opens the OLAP client application. The third option, Batch, allows developers to associate a particular command with a SSAS Action.

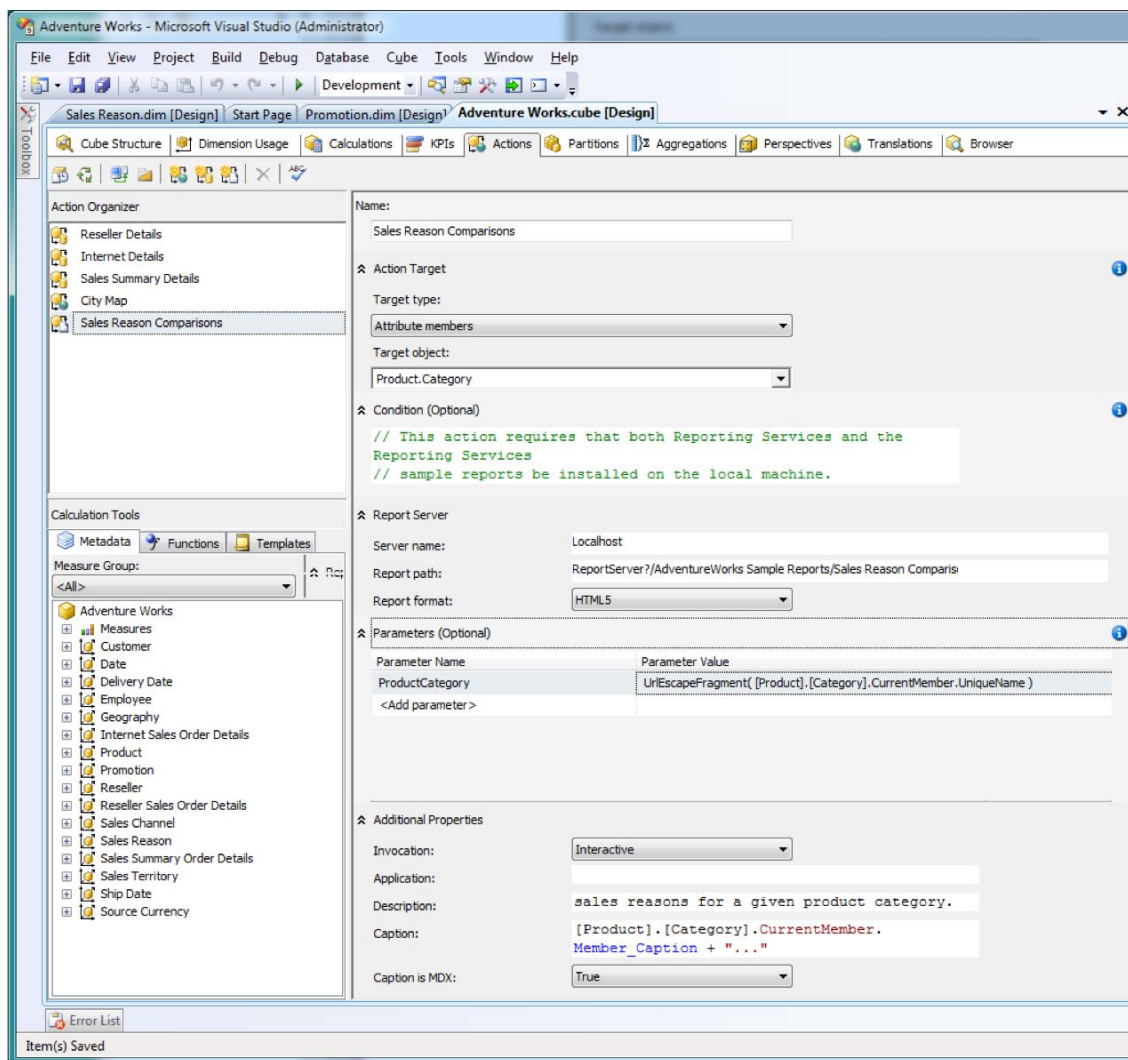


Figure 4-16. Reporting actions allow end users to launch SSRS from their SSAS client applications by right-clicking a cell of interest. A custom URL is generated and passed to SSRS.

Creating Actions in SSMS

Creating an action in SSMS is very similar to creating a KPI, which you did earlier in this chapter. In this section, you will create a Google Reseller Search action sample. The Google Reseller Search action will load Internet Explorer and pass the reseller name as a parameter to www.Google.com. Take note of the Invocation clause, which, when set to Interactive, relies on your end user to execute the action in the

client application. To begin, create a new MDX query for the Adventure Works cube; then enter and execute the following MDX:

```
Create Action [Adventure Works].[Google Reseller Search]
For
  [Reseller] Members
As
  'http://www.google.com/search?q=' + Reseller.CurrentMember.Name,
  Type = URL,
  Invocation = Interactive,
  Application = 'IE',
  Description = 'Google this Reseller';
```

To drop your action when you are completed working with it, simply enter and execute the following MDX:

```
Drop Action [Adventure Works].[Google Reseller Search];
```

Summary

This chapter covered the processes for adding more power to your cube. We reviewed the processes for adding KPIs, translations, perspectives, and actions. Remember that several of these features require the Enterprise Edition of SSAS (and the Enterprise AdventureWorks 2008 R2 samples) as noted in this chapter. The other critical consideration with these features is to verify that all end-user client applications support whatever features you want to add to your cube.

In Chapter 5, we'll delve even deeper into the complexities of SSAS cube modeling. We'll discuss several features, such as the use of multiple fact tables and advanced dimension types (including many-to-many dimensions). We'll also take a more detailed look at the Business Intelligence Wizard.



Advanced OLAP Modeling with SSAS

In this chapter, we'll continue to use the AdventureWorks DW 2008 sample to cover the modeling techniques available to you in SSAS. The chapter will cover the following topics:

- Using multiple fact tables in a single cube
- Modeling nonstar dimension types, including many-to-many (using intermediate fact tables), fact (degenerate), role-playing, and writeback performing advanced dimension modeling, including modeling for either slowly or rapidly changing dimensions and adding custom error handling
- Using the Business Intelligence Wizard for cubes and dimensions to easily enable writeback, semiadditive measures, account/time/dimension intelligence, unary operators, custom member formulas, and attribute ordering.

Multiple Fact Tables in a Single Cube

Beginning with SSAS 2005, SSAS is no longer limited to basing your cube on a single source fact table. Since a single cube can now be based on multiple fact tables, OLAP modeling with SSAS more closely aligns with the way most customers want to view their enterprise data. Much of the work to incorporate multiple fact tables into one enterprise cube needs to be done during the OLAP modeling phase of your project. Conceptually, you can think of this modeling as multiple star schemas, where multiple fact tables reuse the “points” of the stars (or the dimensions). Figure 5–1 illustrates this concept.

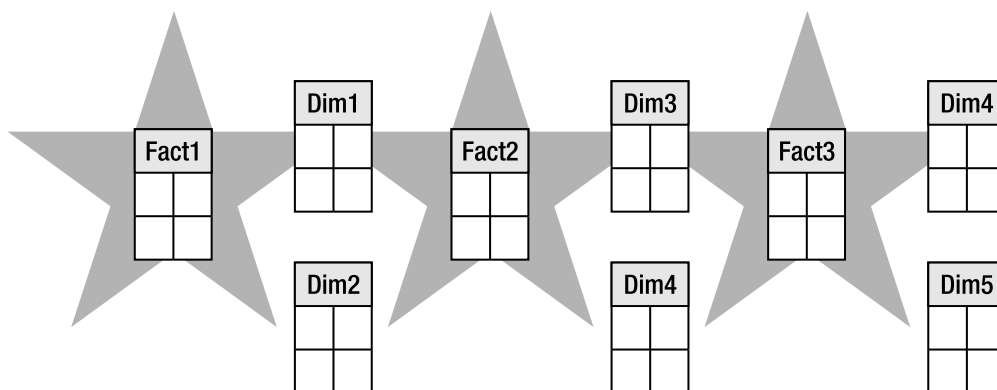


Figure 5–1. You can visualize using multiple fact tables by thinking of multiple star schemas with shared “points” or dimensions.

This type of modeling provides your end users with a true enterprise view of as much business data as needs to be added to a single, all-encompassing cube. From a logical modeling perspective, it makes sense to begin with the single cube model for all BI projects. One factor that could cause you to implement more than one cube has to do with management and administration, namely physical implementation. *Cube partitions* are used to divide cube-processing times and other management tasks, such as security. If you are working with the Enterprise Edition of SSAS, you can create physical partitions for your cube to better manage administration. However, the Standard Edition of SSAS, which has more limited data warehousing features, cannot create any type of physical cube partitions. The mechanics of doing this will be covered in Chapter 6, “Cube Storage and Aggregation.”

When using multiple fact tables in your cube, an important consideration is to what level of granularity you plan to load the rows in each fact table. For example, your company might plan to measure both sales facts (such as sales amount, sales quantity, and so on) and marketing facts (such as promotion cost, promotion audience size, and so on). The sales facts may need to be loaded to the day level of the time dimension in one fact table, and the marketing facts may only need to be loaded to the month level from that same dimension in another fact table. Both of these fact tables can be part of the same cube.

The Dimension Usage tab of the cube designer work area in BIDS is the place where you adjust the granularity of relationship between the dimensions and measure groups. A *measure group* is a container for one or more measures in a cube. Figure 5–2 shows the AdventureWorks sample cube and whether relationships between various dimensions and measure groups exist for the cube being viewed.

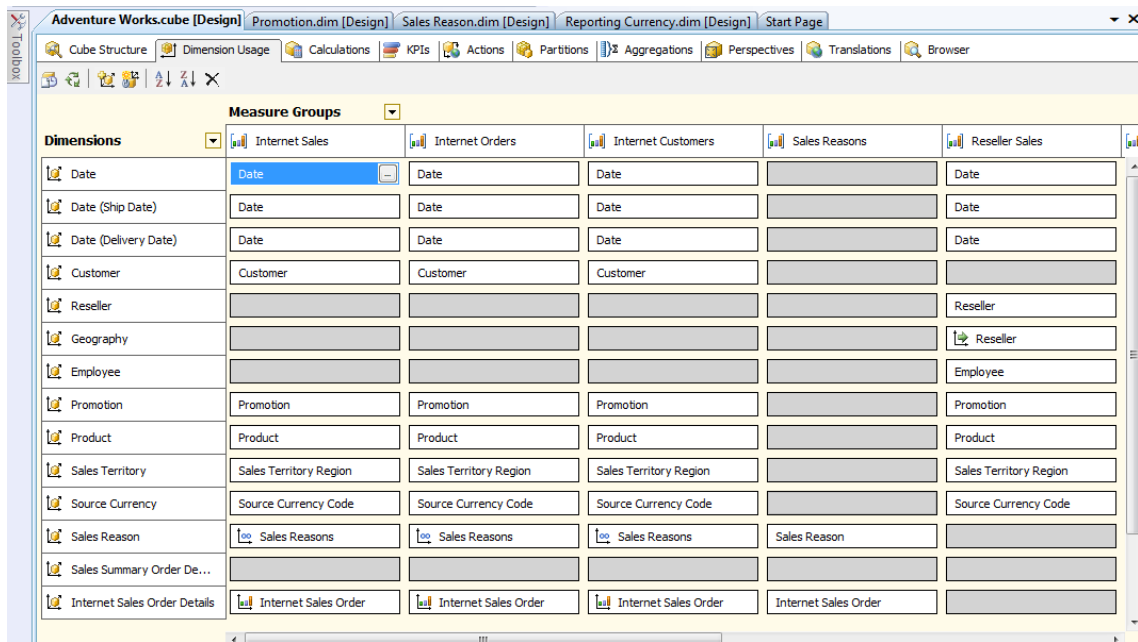


Figure 5–2. The Dimension Usage tab shows the relationship between dimensions and measure groups for the sample AdventureWorks cube. Blank (gray) rectangles indicate that there is no relationship between a particular dimension and measure group.

To examine the granularity of a particular relationship, you click the white rectangle that forms the intersection between the measure group and the dimension. A small gray Build button (with three dots) becomes available on the intersection rectangle. Click the Build button to open the dialog box that allows you to view (or adjust) the type and level of relationship between the Dimension and Measure Group. You'll note that certain dimensions have no relationship to certain fact tables (the intersection is an empty grey rectangle).

In addition to being able to include multiple fact tables in your cube as the basis for measure groups (which contain measures), there is another reason to include more than one fact table in your cube. Interestingly, this last type of fact table does *not* provide additional measures for the cube; rather, it's used for a new type of dimensional modeling called a many-to-many dimension. This is covered in greater detail in the "Modeling Nonstar Dimensions" section later in this chapter.

Nulls

The OLAP community harbors differing opinions about whether or not it is appropriate to load null data into the fact (or dimension) tables of a cube. One of the places you can configure null value handling (for measure groups that intersect with regular [or star] dimensions only) is in the Dimension Usage tab of the cube designer in BIDS (refer to Figure 5–2). To do so, click the Build button on the white rectangle at the intersection of the dimension and fact table that you want to customize. This opens the Define Relationship dialog box, shown in Figure 5–3.

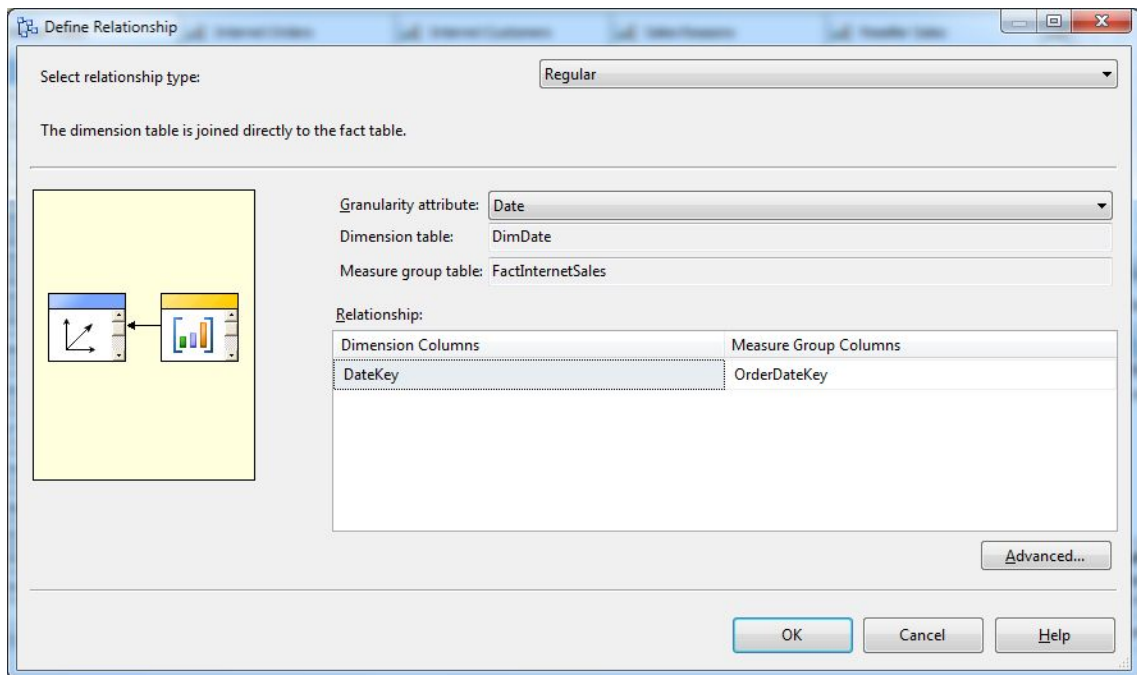


Figure 5–3. The Define Relationship dialog box Advanced option allows you to customize null processing behavior for a particular dimension and fact table.

Tip As a general rule, you should refrain from loading nulls into dimensions and measures, because null values are usually not meaningful to end users. It is typical to test for null values (and to either reject data with null values as error data or to convert the null values to either a number, usually zero, or to a string, for example, “unknown”) and to convert null values to data values during the ETL process of your BI project. This approach is preferred to loading null values into your SSAS cube, because nulls can cause erroneous or unexpected results with MDX aggregate functions.

Clicking the Advanced button in the Define Relationship dialog box takes you to the Measure Group Bindings dialog box (see Figure 5–4). Here, you can set the null-processing behavior to one of the following settings by clicking the Null Processing column value in the Relationship section. Each setting is described here:

- *Automatic*: Converts the null value to zero (for numbers) or an empty string (for strings); the default setting.
- *Preserve*: Keeps the null value.

- *Error*: Raises a null key error; the setting of the `NullKeyNotAllowed` property for the measure controls the result. This setting cannot be used with measures.
- *Unknown Member*: Generates an unknown member and raises a null conversion error; the setting of the `NullKeyConvertedToUnknown` property controls the result. This setting cannot be used with columns associated with measures.

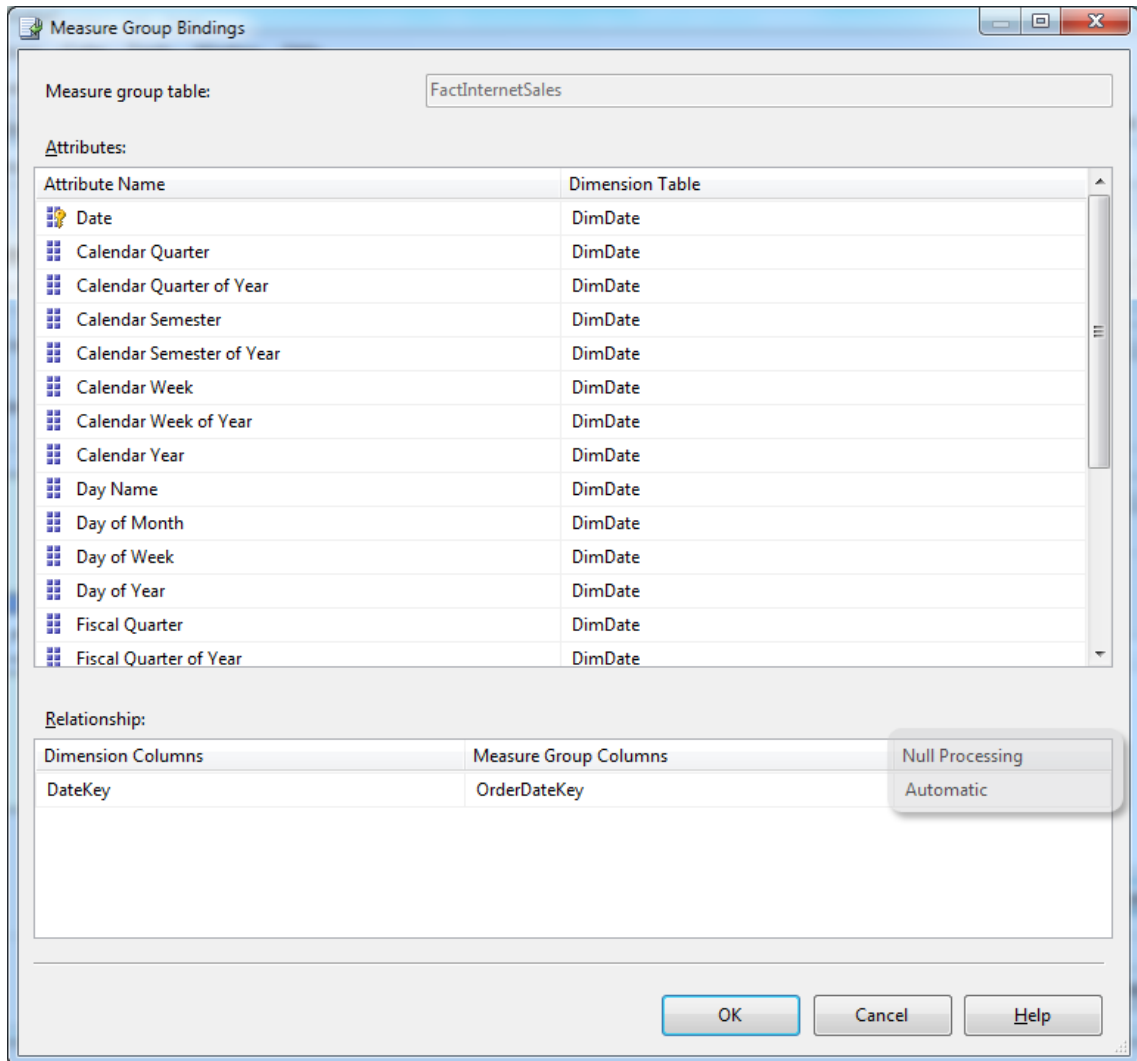


Figure 5–4. Clicking the Null Processing value in the Relationship section of the Measure Group Bindings dialog box allows you to customize null processing behavior for a particular dimension and fact table.

■ **Note** To set the `NullKeyNotAllowed` and `NullKeyConvertedToUnknown` properties in the BIDS Process Dimension (or cube) dialog box, you click the Change Settings button on the Dimension Key Errors tab. Dimension- and cube-processing options are discussed in more detail in Chapter 6.

Nonstar Dimensions

You will probably be able to model the majority of your dimensions using the standard star schema. As SSAS gives you the flexibility to model cube measures by basing them on multiple source fact tables, it also gives you the ability to model dimensions in configurations other than simple star schemas. The types supported are listed briefly here and explained more fully in the following sections:

- *Snowflake*: This dimension is based on more than one table. This is called a referenced dimension in BIDS.
- *Degenerate*: This dimension is based on a column value from one of the fact tables, rather than using a dimension table. This is called a fact dimension in BIDS.
- *Parent-child*: This dimension is based on a hierarchy derived from a self-referencing relationship within a single dimension table. This is modeled using a regular dimension type in BIDS.
- *Many-to-Many*: This dimension is based on two source tables with an intermediate fact table that establishes the many-to-many relationship between the dimension row values. This is called a many-to-many dimension in BIDS.
- *Role playing*: This dimension relates rows in its table to rows in the fact table multiple times. This is most commonly done for the time dimension. An example would be relating fact rows via an order date, a sales date, and a delivery date. This is modeled as a regular dimension in BIDS.
- *Writeback*: This dimension allows authorized end users to update or make changes to the dimensional attribute values but not the dimensional structure, that is, the dimension name, hierarchy structure, level names, and so on. This option requires the Enterprise Edition of SSAS.
- *Mining Model*: This dimension is based on a SSAS Data Mining model (covered in Chapter 14).

Now that you have an overview of the types of dimensions available, the following sections provide examples of when and how to use each type of dimension.

Snowflake Dimensions

Snowflake dimensions use more than one source table. There must be a key relationship between the rows in each table. For example, a Product dimension might use separate Subcategory and Category source tables. Snowflake dimensions add to cube-processing times, so there should be a business reason to use them—the most compelling of which is when you are modeling an entity for which you have a huge number of rows and discrete attributes by subtype. An example of this could be a products dimension with product category-specific attributes. In other words, product category 1 has a discrete

group of attributes; product category 2 has a different group of attributes; product category 3 has yet another group of attributes, and so on.

For example, if your business scenario is to model product sales for a discount chain, you might need to model these three product categories: makeup, paint, and shoes. The makeup category may have attributes such as “skin type” or “product base type,” which are not meaningful to the other product categories. If you used a star design for this example, you could create a needlessly large dimension due to the amount of nulls in the star schema’s source dimension table or tables.

Referenced dimensions, like all dimension types, are configured in the BIDS cube designer in the Dimension Usage tab. To configure the relationship between the rows in a particular dimension and fact table, click the gray rectangle that intersects the dimension and fact table. Then, click the Build button (the icon with three dots) to open the Define Relationship dialog box. For a snowflake, you select Referenced dimension from the Select relationship type drop-down list. The Geography Dimension in the AdventureWorks sample provides an example of a snowflake dimension as shown in Figure 5–5.

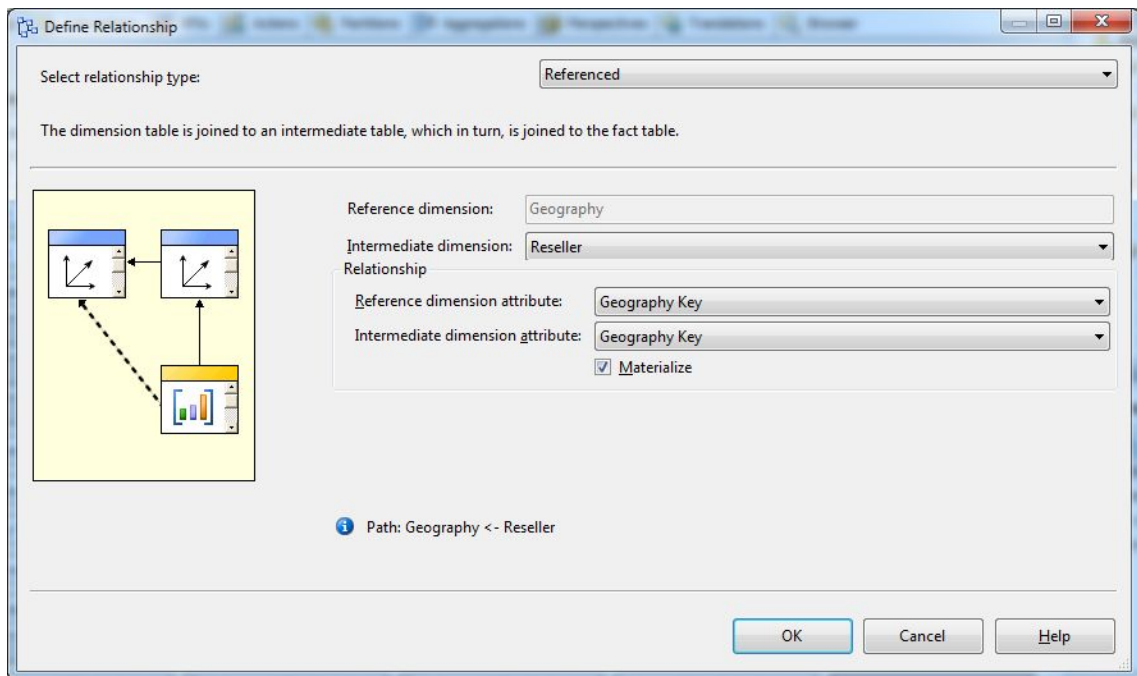


Figure 5–5. The Define Relationship dialog box in BIDS allows you to set up snowflake (or referenced) and many other types of dimensions.

■ **Tip** Leaving the default Materialize check box checked is recommended. This speeds up queries by requesting that SSAS store the value of the link between the dimensional tables and fact table as MOLAP metadata (meaning that the metadata is stored on SSAS in a multidimensional format) when the cube or dimension is processed.

Degenerate Dimensions

Degenerate dimensions (called *fact* dimensions in BIDS) use a column in the fact table as a source, as opposed to an actual dimension table. For example, a Sales Order Detail dimension might have an attribute called PO Number. Defining a dimension-to-fact relationship using BIDS follows the same procedure listed in the previous section, “Snowflake Dimensions.” After you’ve accessed the Define Relationship dialog box, you then select the Fact relationship type. Figure 5–6 shows an example from AdventureWorks.

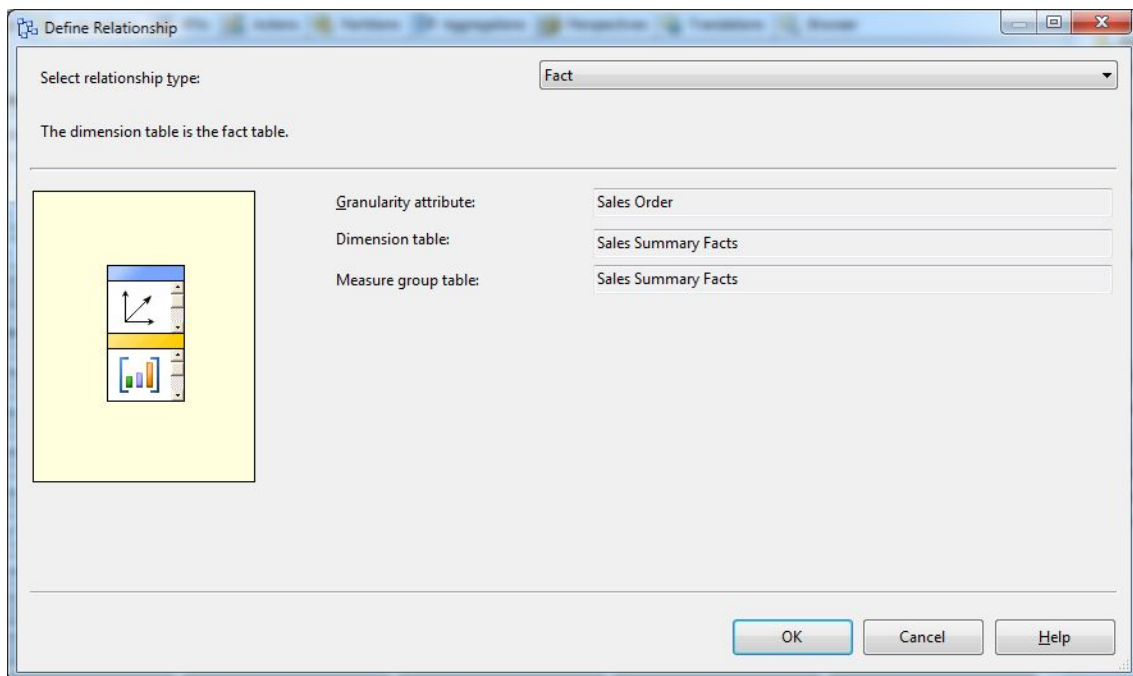


Figure 5–6. Select the Fact relationship type to create a relationship between a dimension based on a column in the fact table and the fact table itself. This type of dimension is also called a degenerate dimension.

When you are modeling this type of dimension, you should carefully consider the percentage of end users that will need to browse the attributes included in any fact dimension. These values add to the size of your fact table and can, in turn, add to cube-processing times. This may also result in slower queries. Degenerate dimensions, used as a track-back to a system of record, can aid end users that use multiple systems in their daily work. However, if your analysis determines that this attribute belongs in a dimension, create a separate dimension source table and a regular dimension for this type of information.

Parent-Child Dimensions

Parent-child dimensions are derived from a single source table. This table must have a primary key and a foreign key that allows a self-join to be performed on this same table. An example from AdventureWorks cube is the employee table. This type of dimension is used to create a self-referencing, ragged hierarchy. Another way to think of this is a hierarchy that does not necessarily have a larger number of leaf-level members below its parent level, or in the case of the employee table, some employees are managers, or have direct reports, and some are not. Parent-child dimensions are called *Regular* dimensions in the Define Relationship dialog box. An example using the employee table is shown in Figure 5–7.

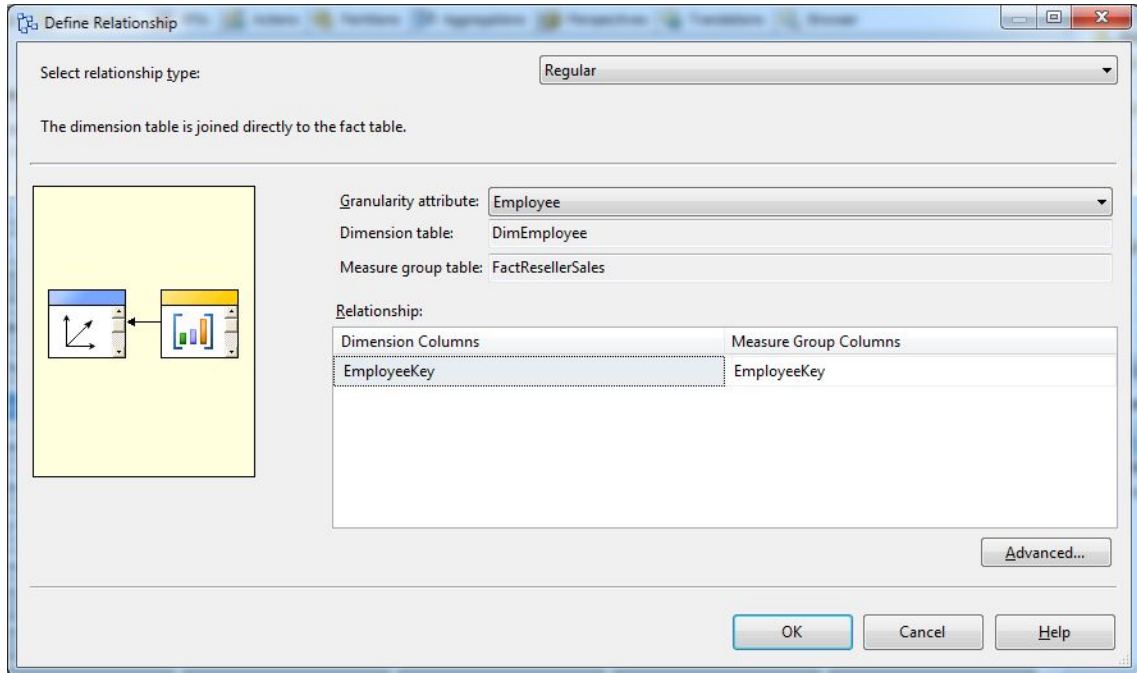


Figure 5–7. Parent-child dimensions have relationships to the fact table of type *Regular*.

You can verify that the key relationship was correctly detected in the Data Source View (DSV) for the cube. Using the sample, AdventureWorks, you'll note that the employee table shows this self-referential relationship (see Figure 5–8).

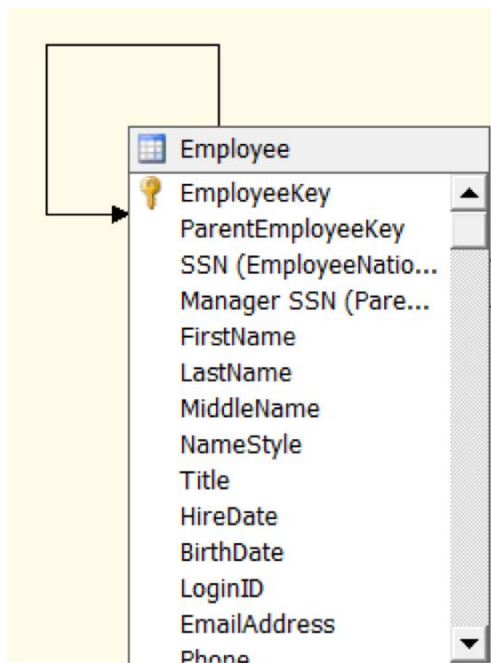


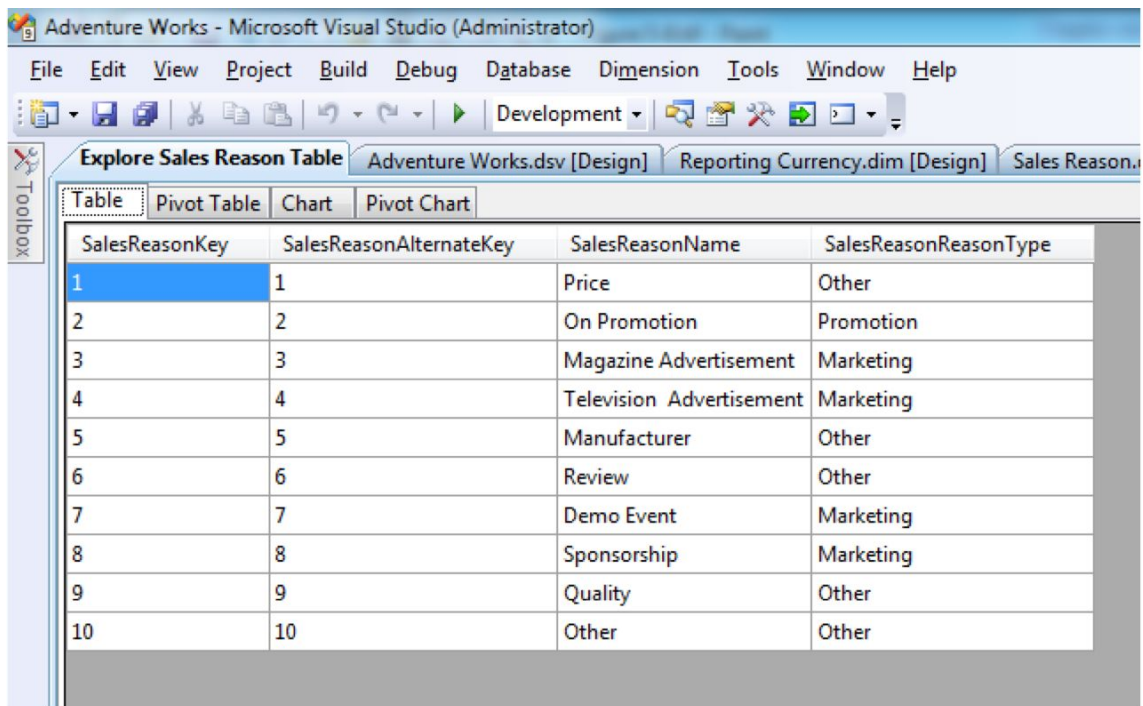
Figure 5–8. Parent-child relationships can be verified in the DSV for the cube.

Another way to verify that you've set up the parent-child relationship correctly is to browse the dimension in BIDS, where the source columns will be shown as attributes. You'll review the Usage property for the columns listed as attributes. The three possible values are Parent, Key, and Name. In this example, BIDS will generate an attribute named Employee with a Parent Usage, an attribute called EmployeeKey with Key Usage, and other attributes for all other columns in the source table (with the same name as the source columns) with a Name Usage.

Many-to-Many Dimensions

The many-to-many dimension involves two source dimension tables, plus an intermediate table, which makes a total of at least three source tables. At least two of these tables are typical dimension tables. The relationship between the rows in the second dimension table and the main fact table is established via keys in those two tables.

You can think of the secondary fact table as similar to a relational junction table in OLTP modeling. An example of this is provided in the AdventureWorks sample for the salesReason dimension. The business case is that each Internet sale can have many sales order lines; each order line can have a sales reason. So that each Internet sale can have many sales order lines and each order line has a sales reason, the relationship between order lines and sales reasons is many-to-many. The first step to using many-to-many dimensions is modeling and creating the appropriate source tables, as shown in Figures 5–9 and 5–10.



| SalesReasonKey | SalesReasonAlternateKey | SalesReasonName | SalesReasonReasonType |
|----------------|-------------------------|--------------------------|-----------------------|
| 1 | 1 | Price | Other |
| 2 | 2 | On Promotion | Promotion |
| 3 | 3 | Magazine Advertisement | Marketing |
| 4 | 4 | Television Advertisement | Marketing |
| 5 | 5 | Manufacturer | Other |
| 6 | 6 | Review | Other |
| 7 | 7 | Demo Event | Marketing |
| 8 | 8 | Sponsorship | Marketing |
| 9 | 9 | Quality | Other |
| 10 | 10 | Other | Other |

Figure 5–9. Many-to-many dimensions require two source dimension tables, plus an intermediate fact table (that functions much like a relational junction table). One of the tables lists the dimensional attribute value. In the AdventureWorks example, this is the SalesReason table.

Adventure Works - Microsoft Visual Studio (Administrator)

File Edit View Project Build Debug Database Tools Window

Development

Explore Internet ...Reason Facts Table Adventure Works.dsv [Design]

Table Pivot Table Chart Pivot Chart

| SalesOrderNumber | SalesOrderLineNumber | SalesReasonKey |
|------------------|----------------------|----------------|
| SO43697 | 1 | 5 |
| SO43697 | 1 | 9 |
| SO43702 | 1 | 5 |
| SO43702 | 1 | 9 |
| SO43703 | 1 | 5 |
| SO43703 | 1 | 9 |
| SO43706 | 1 | 5 |
| SO43706 | 1 | 9 |
| SO43707 | 1 | 5 |
| SO43707 | 1 | 9 |
| SO43709 | 1 | 5 |
| SO43709 | 1 | 9 |
| SO43710 | 1 | 5 |
| SO43710 | 1 | 9 |
| SO43711 | 1 | 5 |
| SO43711 | 1 | 9 |
| SO43712 | 1 | 5 |
| SO43712 | 1 | 9 |
| SO43713 | 1 | 5 |
| SO43713 | 1 | 9 |
| SO43714 | 1 | 5 |
| SO43714 | 1 | 9 |
| SO43715 | 1 | 5 |
| SO43715 | 1 | 9 |

Error List

Ready

Figure 5–10. Many-to-many dimensions require at least three source tables. One of the tables establishes the relationship between the dimension table attributes and the rows in the destination fact table. In the AdventureWorks example, this is the FactInternetSalesReason table.

You can also see whether you've modeled the relationship correctly by reviewing the participating tables in the DSV as shown in Figure 5–11.

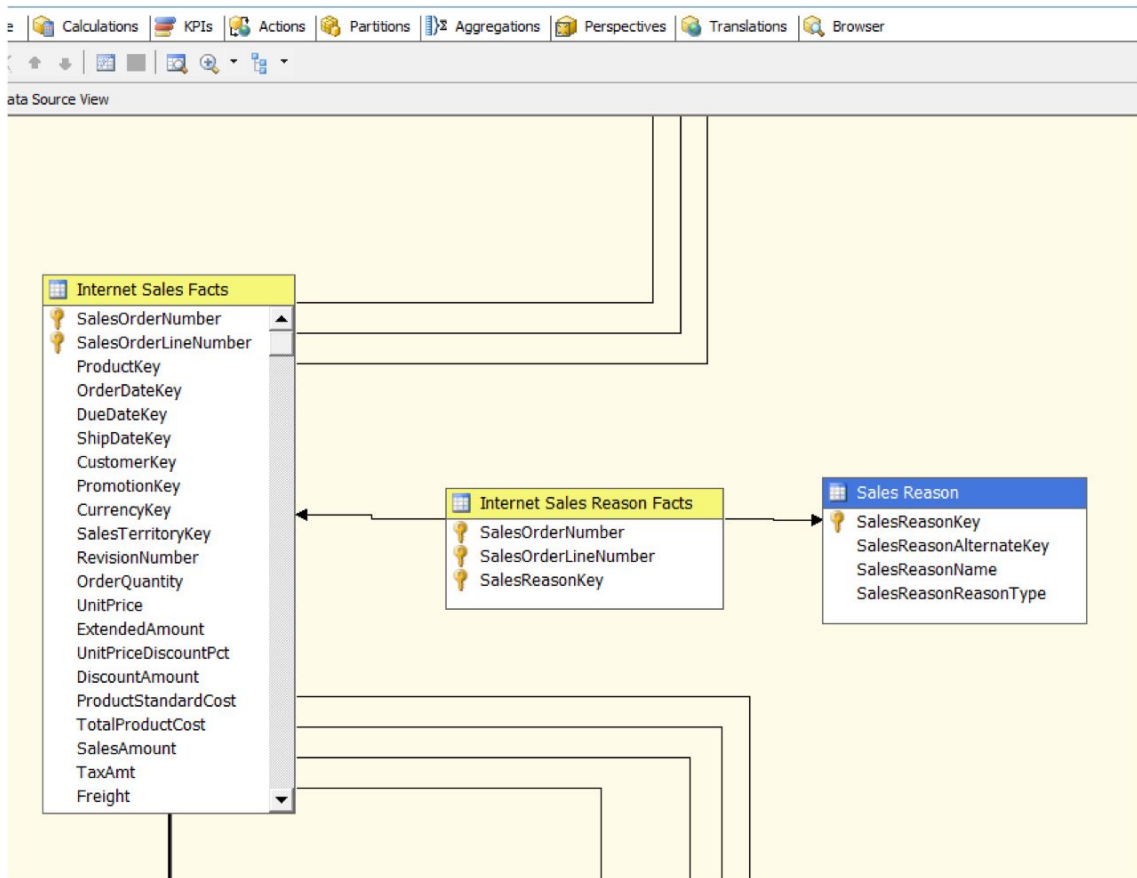


Figure 5–11. You can verify that you've modeled the many-to-many relationship in the DSV for your cube. The intermediate table will appear as an additional fact table in this view. A portion of the AdventureWorks sample is shown here.

After you've correctly modeled the many-to-many dimension, you establish the relationship using the Dimension Usage tab of the cube designer. While working in the Define Relationship dialog box, you select the Many-to-Many dimension type, as shown in Figure 5–12.

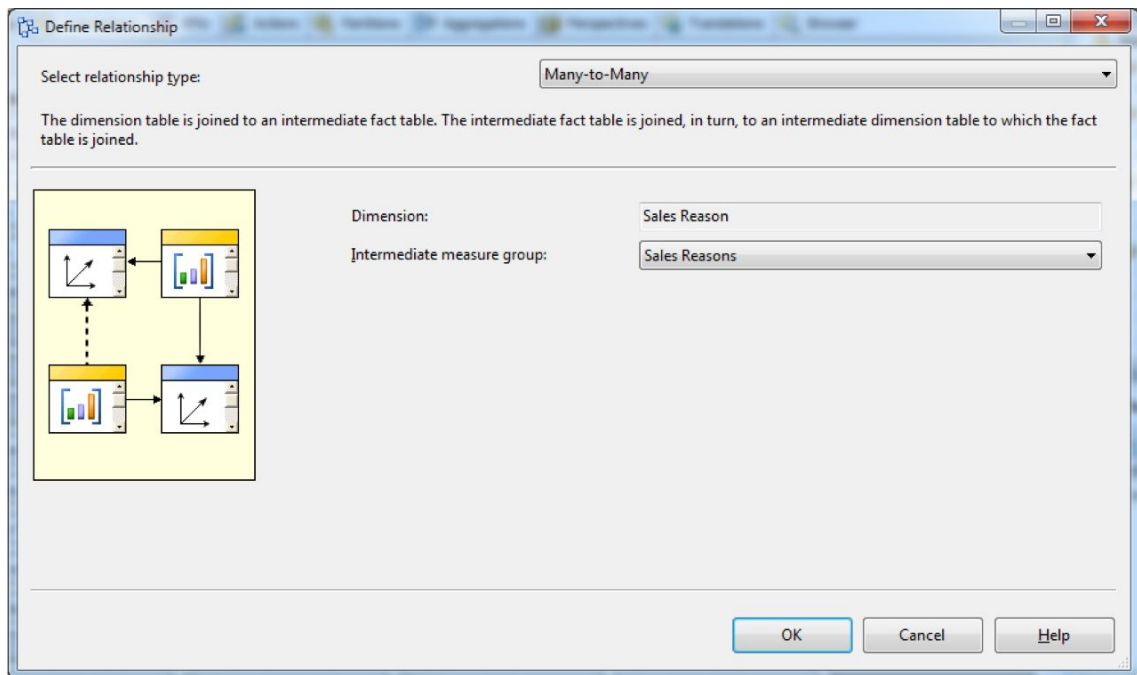


Figure 5-12. After selecting the *Many-to-Many* relationship type in the *Define Relationship* dialog box, you must select a table in the *Intermediate measure group* list box.

Role-Playing Dimensions

Role-playing dimensions are set up via modeling as well. A typical example from the AdventureWorks sample (from the time dimension) is shown in Figure 5-13—the three instances of the Date dimension in the Dimension Usage tab. Role playing means joining rows from a dimension to a fact table more than once. The example for this case is Date, Ship Date, and Delivery Date. To do this, you join more than one dimension based on the same source table by simply adding (or referencing) the same source table multiple times as the basis for cube dimensions and giving each dimension a different dimension name. In other words, you add new dimensions based on the same table multiple times. The example from AdventureWorks is the most typical implementation of a role-playing dimension, adding a time dimension multiple times to create different time-based dimensions.

Figure 5-13 shows the modeling of this type of dimension in the Dimension Usage tab of the cube designer in BIDS.

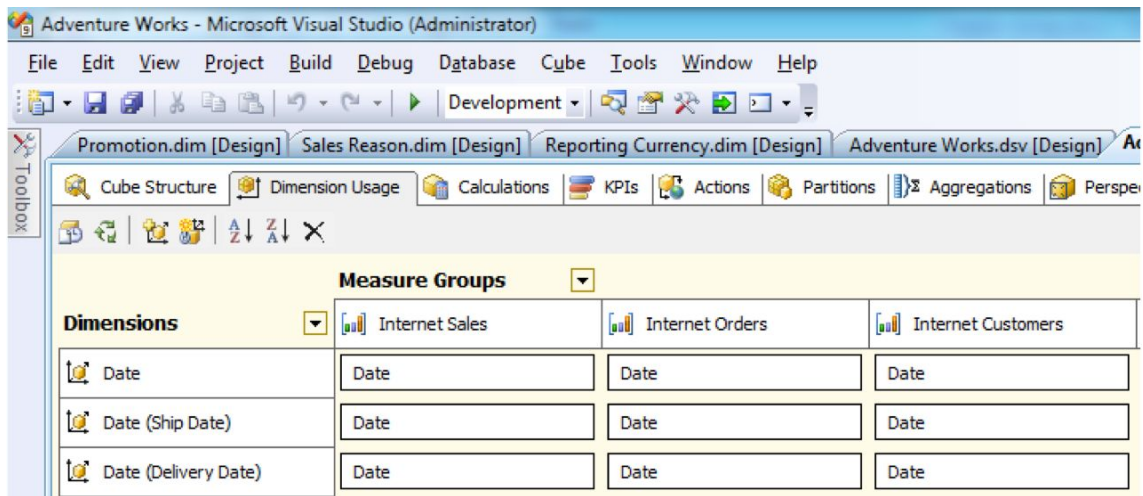


Figure 5–13. Role-playing dimensions are modeled as regular dimensions in the Dimension Usage window. You model this type of dimension by creating multiple dimensions based on the same source table. A typical use of this is with a time dimension.

Writeback Dimensions

Writeback dimensions allow you to add, update, and delete dimension values in any dimension that is based on a single table. Writeback values are kept in a special table that is added to your data source. Any writebacks you perform are immediately available for viewing in your cube.

Writeback dimensions require the Enterprise Edition of SSAS. You enable writeback for a dimension by setting the `WriteEnabled` property value of a dimension to `True`. You can also use the Business Intelligence Wizard to enable writeback for a dimension. Figure 5–14 shows the property value interface for a dimension to enable writeback for an entire dimension (which means all attributes at all levels). You cannot enable writeback for any subset from a dimension, which means that you cannot enable writeback for a particular attribute of a dimension. Writeback is an “all or nothing” option for a particular dimension.

With writeback dimensions, you need to verify that your selected client applications support writeback. Also, you must confirm that allowing writeback is consistent with your project’s business requirements. Another consideration is that you must specifically grant read/write permissions to write-enabled dimensions for those end users who will need to write to a dimension. In my experience, writeback is not commonly enabled for BI projects. One case where it may be enabled is when a cube is used for financial forecasting, particularly “what if” scenarios.

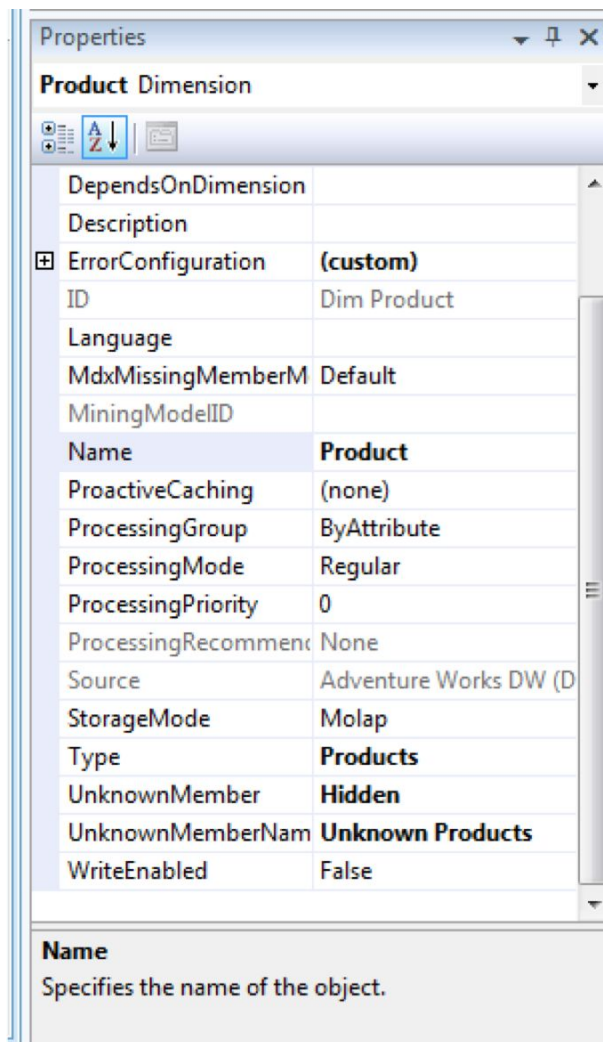


Figure 5–14. You can write enable dimensions by setting the `WriteEnabled` property of an entire dimension to `True`.

■ **Note** Writeback is *not* supported for certain dimensions types, including referenced dimensions, fact dimensions, many-to-many dimensions, and linked dimensions. As previously mentioned, you can only enable writeback for entire dimensions, not specific attributes of a dimension.

Dimensions That Change

After you've selected the type of dimension to be used for your particular business scenario, another important concept is whether and how your model will support update behavior for the dimensional attribute values. To understand how to model for updates, consider that updates in SSAS need to be divided into two types: new values (or inserts) and changes (or updates or deletes of existing attribute values). Allowing inserts of new values for the dimension or fact tables does *not* require you to use any special modeling techniques. However, the business rules required for *changes* to dimensional attributes *will* affect the way you model those attributes.

In Chapter 2, we introduced the topic of modeling slowly (or rapidly) changing dimensions. At this point in your BI solution, you now have six considerations. These considerations include modeling for the rate of change (that is, slow or rapid), and configuring error definitions (that is, "Are changes allowed at all?" or "Are nulls allowed?"). The options and support for updating the various types of dimensions via the SSIS slowly changing dimension transformation are listed next. The reason for this last point is that you will probably want to automate the process of dimensional attribute updates via SSIS packages.

The considerations are as follows:

- *Slowly Changing Type 1:* Any submitted changes overwrite previous values. The SSIS "slowly changing dimension transformation" calls this type changing. If you choose this method, you may want to add a column to your dimension table called *dateLoaded*, if your business requirements call for this.
- *Slowly Changing Type 2:* Any submitted changes will result in a new record (or row) being added. The SSIS "slowly changing dimension transformation" calls this type historical. The structure of your dimension tables will need columns for start date and end date. Additionally, you can add a *current* or *active* record flag as well.
- *Slowly Changing Type 3:* Adding more attributes (or column values) when the dimension member value changes. The SSIS "slowly changing dimension transformation" does not support this type. The structure of your dimension tables must accommodate this. This type requires a source table structure, which includes as many columns (and corresponding date effective columns) for the number of changes you want to capture.
- *Rapidly Changing:* The dimensional attributes change constantly, for example, fast-food restaurant location employee names. In this case, you would usually process your dimension differently altogether. If your business requirements include a rapidly changing dimension, you can use ROLAP (or relational OLAP storage).

This means that the dimension metadata is not copied to SSAS; rather, it is read "live" from the star schema. This option is discussed in more detail in Chapter 6. This type of dimension does not require any special modeling. You simply set the *storageMode* property of the dimension to ROLAP. Be aware that this can result in significantly slower query processing. This option also requires the Enterprise Edition of SSAS.

- *No changes allowed:* Requests for changes are treated as errors (and are usually logged as errors). The SSIS "update slowly changing dimension transformation" calls this type fixed. You configure a custom error configuration to support this business requirement. This is discussed in more detail in the next paragraph.

- *Nulls allowed:* If you choose to allow null values on load, then you should establish a standard method of “handling” or renaming those null values. This is configured via the UnknownMember dimension property. The SSIS “update slowly changing dimension transformation” calls this type an inferred member. This property is discussed further in subsequent paragraphs of the next section.

Error Handling for Dimension Attribute Loads

If your business needs require more granular control over error handling, you can set several properties in the SSAS dimension editor, at the dimension level. First, let’s consider what types of errors could occur. These could include loading nulls, loading mismatched (to the rows in the fact table) key values, and loading duplicate key values. You should capture desired error-handling behavior for each dimension during the requirements gathering phase of your BI project.

The first consideration when implementing those requirements is whether you’ll need to include a custom error configuration. If you set the ErrorConfiguration property to (custom), several properties become available for you to configure in BIDS via the dimension property sheet. These properties are shown in Figure 5–15; all the values are the defaults, with the exception of the KeyErrorAction property value, which we have changed from the default value of ConvertToUnknown to the optional value of DiscardRecord for this example.

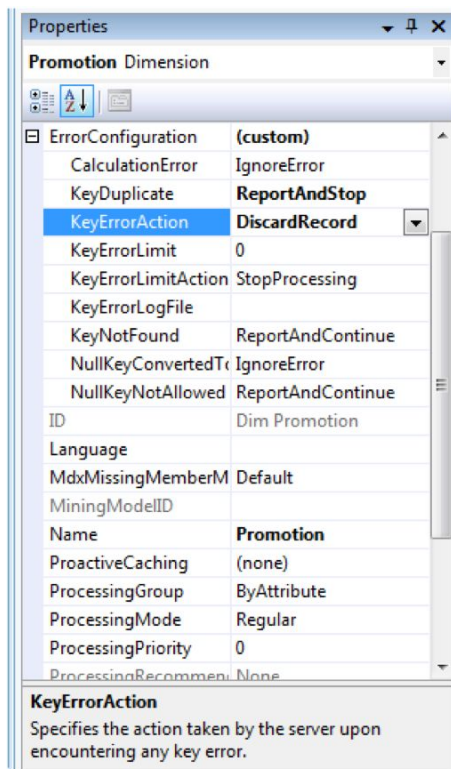


Figure 5–15. You set the behavior of key errors at the level of the entire dimension in SSAS.

Along with custom error handling, you may want to control the value of nulls loaded into a dimension by changing the `UnknownMember` property of the dimension to a nondefault value for a dimension, as shown in Figure 5–16.

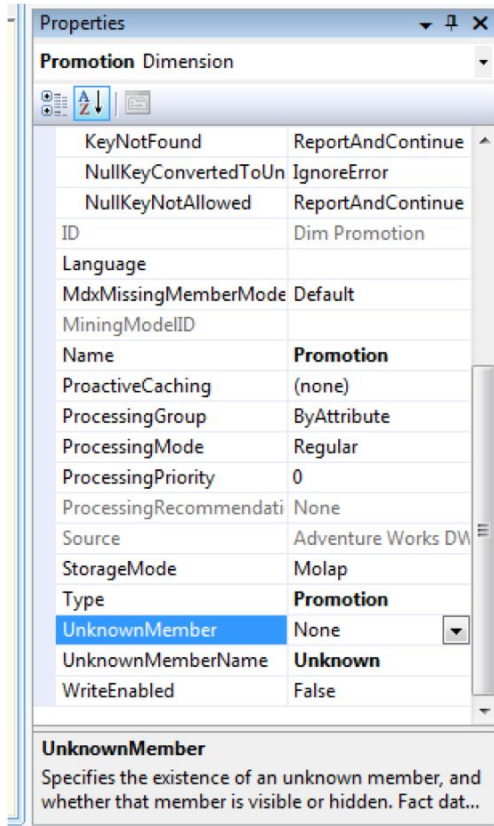


Figure 5–16. The `UnknownMember` property of a dimension allows you to control the behavior of nulls (if you choose to load null values into a dimension).

It is still best practice to cleanse all data prior to loading into your cube via the ETL processes (using SSIS). Most of our clients prefer to refrain from loading nulls into both the dimension and fact tables. However, should your business requirements be such that you load nulls, SSAS gives you many options for working with them.

Using the Business Intelligence Wizard

After you've added the fact tables and dimensions to your cube and configured the properties for both, the next option for you to consider is whether to use the Business Intelligence Wizard to add still more sophisticated options to your BI solution. This wizard presents you with a different list of options

depending on whether you've opened it from the cube or from the dimension designer. The complete list of options (for both types) is as follows:

- **Add Writeback:** This option, for dimensions only, requires the Enterprise Edition of SSAS. Using this wizard simply sets the `WriteEnabled` property of a dimension to `True`.
- **Add Semiadditive Measures:** This option, for cubes only, allows you to set the `aggregationFunction` property for a cube measure to something other than the default (Sum). The choices are Average of Children, By Account, Count, Distinct Count, FirstChild, FirstNonEmpty, LastChild, LastNonEmpty, Max, Min, None, and Sum.
- **Add Account Intelligence:** This allows you to assign accounting types, that is, income, expenses, and so on, to dimension attributes. Specifically, you associate one dimension attribute with one of the following: Chart of Accounts, Account Name, Account Number, or Account Type. The default is to set measures to be semiadditive based on account type. The wizard will set the `aggregationFunction` property to the value `byAccount`. The Configure Dimension Attributes page of the wizard is shown in Figure 5–17.

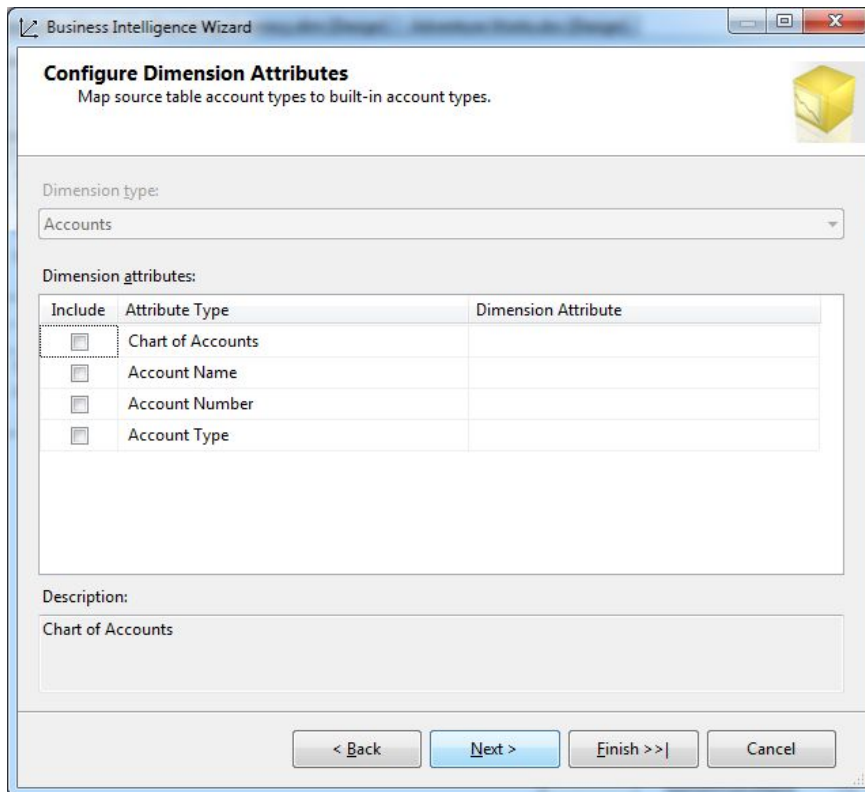


Figure 5–17. In the Business Intelligence Wizard's Add Account Intelligence Configure Dimension Attributes page, you can associate attribute types with specific attributes.

- *Add Time Intelligence*: This option, for cubes only, generates an MDX script that creates calculated members (which appear in the cube as regular measures) based on common time-based comparisons that you've selected in the wizard page.

Some examples of common time-based comparisons are *Quarter to Date*, *Three Month Moving Average*, and *Year Over Year Growth*. Figure 5–18 shows the list of options. Figure 5–19 shows a sample generated MDX script.

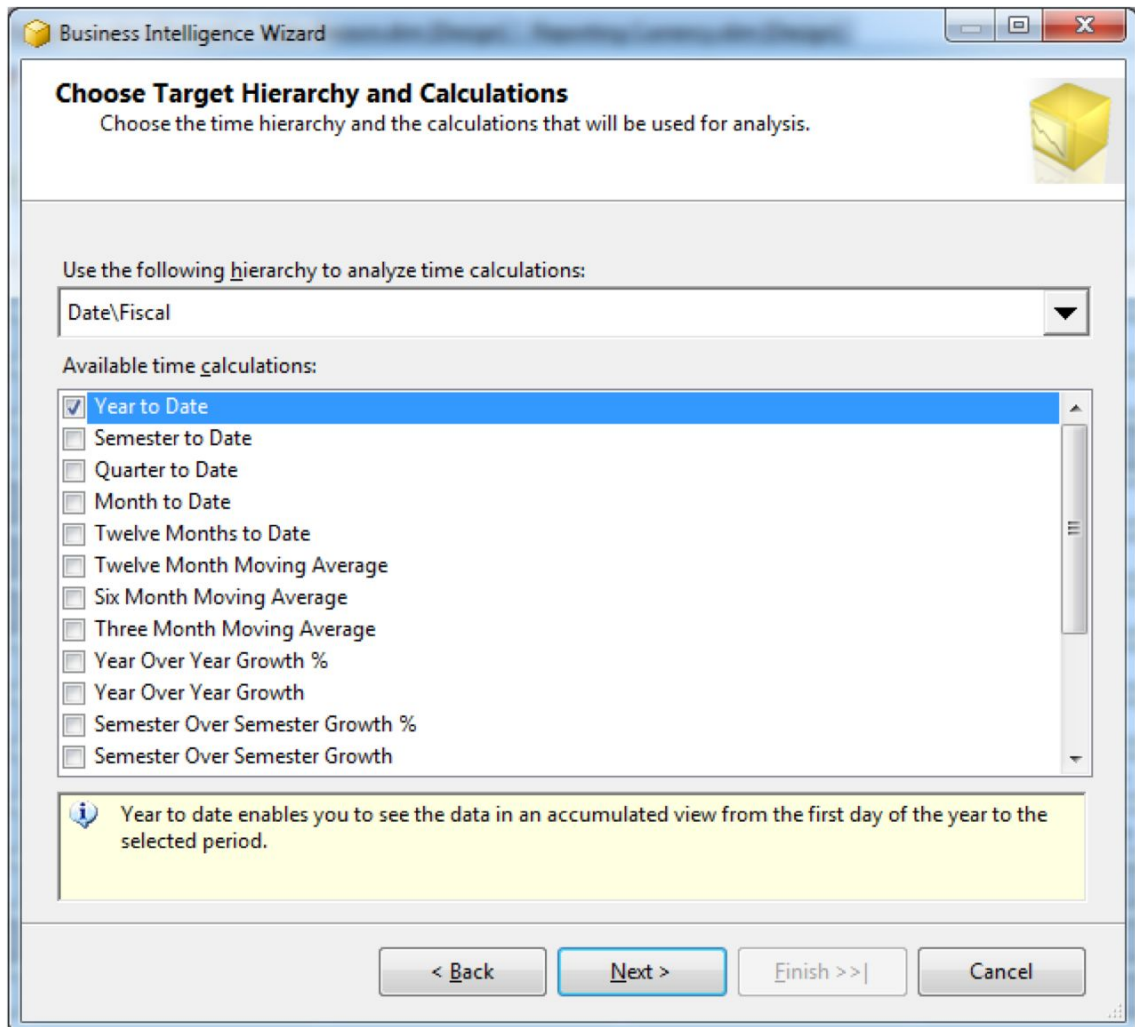


Figure 5–18. In the Business Intelligence Wizard's *Add Time Intelligence* selection, you select common time-based scenarios and SSAS generates an MDX script for you. This script will add calculated members to your OLAP cube.

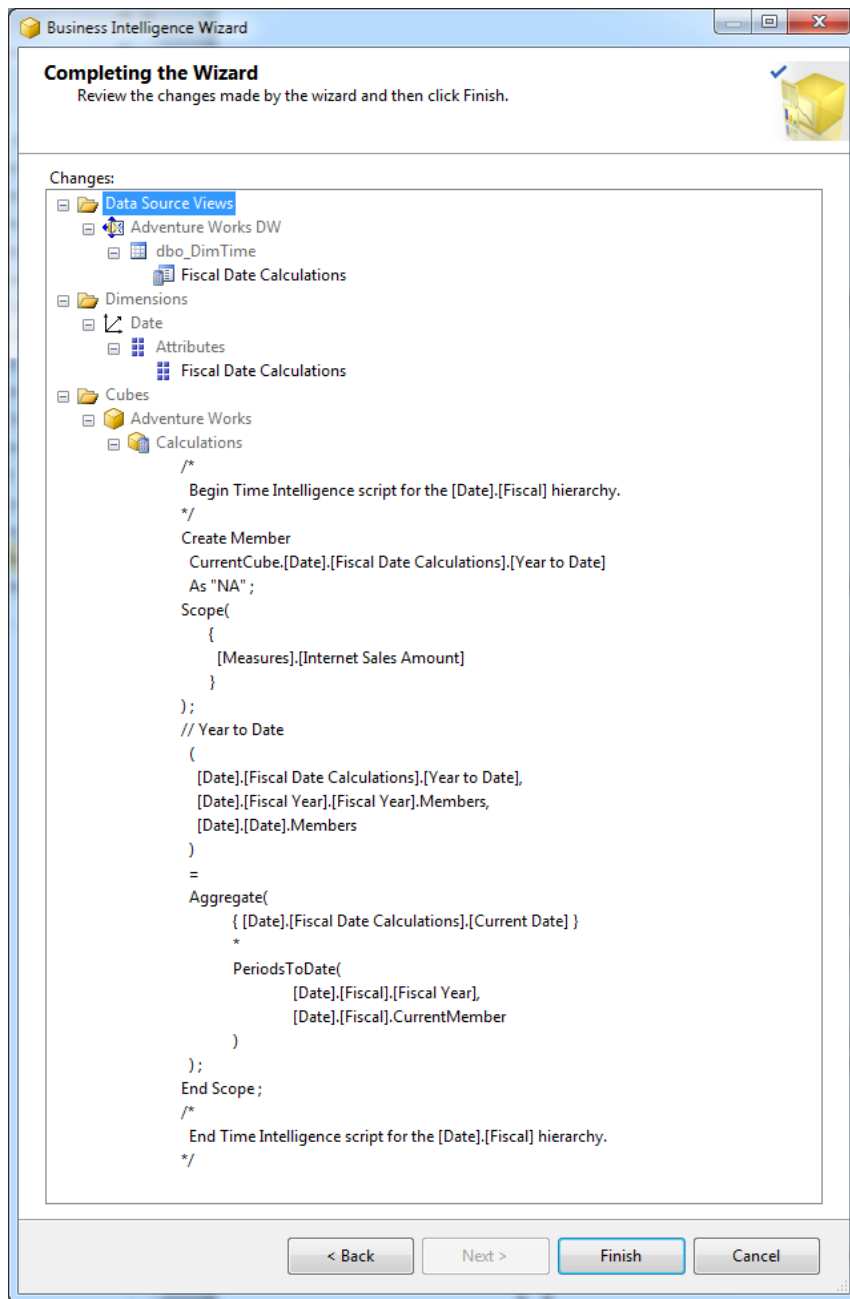


Figure 5–19. In the Business Intelligence Wizard’s Add Time Intelligence selection, SSAS generates a MDX script that creates a calculated member in the cube.

- *Add Dimension Intelligence:* This option allows you to associate metadata with dimension attributes by setting the type property. For example, for the customer dimension, you can associate a particular dimension attribute with the address type, the city type, or the company type. Some client applications use this type information to present information to end users in a more friendly fashion and to assist end users in adding calculated members by using GUI tools included in their particular client application, rather than by having to manually author MDX scripts.
- *Add Unary Operator:* This cubes-only option allows you to specify a unary operator to replace the default aggregation (which is to sum all measures) for noncalculated hierarchy members in parent-child hierarchies. You can specify either regular or weighted aggregations. Like the Add Account Intelligence option, this option is often used when your business requirements include modeling to support an organization's balance sheets.
- *Add Custom Member Formulas:* This one allows you to specify a column with a custom MDX formula that will override the default rollup. The default rollup is to sum all measures. This sets the CustomRollupColumn property of the selected dimensional attribute to the selected column.
- *Add Attribute Ordering:* For dimensions only, this option allows you to override the default setting, "Order by member name," to replace with "Order by key" or "Order by name" of a different attribute in the same dimension.
- *Add Currency Conversion:* This one, for cubes only, allows you to add a dimension that is used to translate one or more currencies in the fact table to one or more new currencies. I covered the implementation of this option by reviewing the steps in the wizard in detail in Chapter 4.

Summary

This chapter covered many advanced OLAP modeling techniques. These methods included using multiple fact tables and various types of dimensions. SSAS has quite a bit of flexibility in its capacity to allow for modeling scenarios that more closely match that of your actual business data. Remember that several of these features require the Enterprise Edition of Analysis Services.

The other important consideration with these techniques is to always use business requirements as a justification for adding the complexity that any of these new capabilities bring to your BI solution. I've seen many BI solutions that were overcomplicated, which made them difficult to maintain, to administer, and most important, to use. Favor simplicity in cube design.

In the next chapter, you'll learn about physical storage mechanisms for cube and dimension data and metadata. This is the world of MOLAP, HOLAP, or ROLAP—multidimensional, hybrid, or relational OLAP. You'll also learn about what exactly is happening when you "process" a cube or dimension.



Cube Storage and Aggregation

Now that you've completed your cube's logical design, it's time to design the physical storage mechanisms, including file placement and aggregations.

This chapter covers the details of physical cube processing and storage, including an explanation of the what, when, and why to use any of the three cube storage methods: MOLAP, HOLAP, or ROLAP. These storage-type acronyms stand for *Multidimensional, Hybrid, or Relational OLAP*. The chapter also explains cube aggregations—what they are and when and when not to use the default cube-processing settings (which use no aggregations). Following is a complete list of topics for this chapter:

- Using default storage—MOLAP with no aggregations
- Designing custom aggregations
- Understanding advanced storage—MOLAP, ROLAP, and HOLAP
- Working with proactive caching
- Designing relational partitions and SSAS partitions
- Planning for rapidly changing (ROLAP) dimensions
- Selecting appropriate cube-processing options

Using the Default Storage: MOLAP

From earlier chapters, you may remember that to view structural changes to your cube, you must process the cube in BIDS and then deploy the changes to Analysis Services. To understand what is happening during this process, you must first be aware of what you are creating when you design and build BI objects, such as DSVs, cubes, and dimensions using BIDS. Although you are working in a GUI environment, commands in the XMLA SSAS metadata language are being generated behind the scenes.

XML for Analysis

XML for Analysis (XMLA) is an open-standard XML dialect that is used for client-server communications between OLAP servers. XMLA describes its messages in a SOAP-style format designed to access multidimensional databases that contain OLAP cubes. Visit <http://www.xmla.org/> to learn more about XMLA. Simple Object Access Protocol (SOAP) is a type of XML dialect. XMLA will sometimes also contain Multidimensional Expressions (MDX) queries, and it is the language of cube metadata. This language includes many methods that allow you to view or modify cube information; two examples are the Discover and Execute methods. For an example of XMLA that uses the Create method, see Figure 6–1.

For a complete list of XMLA methods, see the Microsoft Books Online (BOL) topic “Using XML for Analysis in Analysis Services (XMLA)” at <http://technet.microsoft.com/en-us/library/ms186654.aspx>.

```

XMLAQuery2.xml...2 SE (GFAA\Guy)
<Create xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
  <ParentObject>
    <DatabaseID>Adventure Works DW 2008R2 SE</DatabaseID>
  </ParentObject>
  <ObjectDefinition>
    <Dimension xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
      <ID>Dim Customer</ID>
      <Name>Customer</Name>
      <Annotations>
        <Annotation>
          <Name>http://schemas.microsoft.com/DataWarehouse/Designer/1.0:UseDiagramDefaultLayout</Name>
          <Value>true</Value>
        </Annotation>
        <Annotation>
          <Name>http://schemas.microsoft.com/DataWarehouse/Designer/1.0:ShowRelationshipNames</Name>
          <Value>false</Value>
        </Annotation>
        <Annotation>
          <Name>http://schemas.microsoft.com/DataWarehouse/Designer/1.0:DiagramBoundingTop</Name>
          <Value>-5464</Value>
        </Annotation>
        <Annotation>
          <Name>http://schemas.microsoft.com/DataWarehouse/Designer/1.0:ShowFriendlyNames</Name>
          <Value>true</Value>
        </Annotation>
        <Annotation>
          <Name>http://schemas.microsoft.com/DataWarehouse/Designer/1.0:DiagramBoundingLeft</Name>
          <Value>2456</Value>
        </Annotation>
        <Annotation>
          <Name>http://schemas.microsoft.com/DataWarehouse/Designer/1.0:DiagramViewPortLeft</Name>
          <Value>641</Value>
        </Annotation>
        <Annotation>
          <Name>http://schemas.microsoft.com/DataWarehouse/Designer/1.0:DiagramLayout</Name>
          <Value>
            <dds xmlns="">
              <diagram fontclsid="{0BE35203-8F91-11CE-9DE3-00AA004BB851}" mouseiconclsid="{0BE35204-8F91-11CE-9DE3-00AA004BB851}">
                <font>
                  <ddsxmlobjstreamwrapper binary="01000000900144420100065461686f6d61" />
                </font>
                <mouseicon>
                  <ddsxmlobjstreamwrapper binary="6c74000000000000" />
                </mouseicon>
              </diagram>
              <layoutmanager>
                <ddsxmlobj />
              </layoutmanager>
              <ddscontrol controlprogid="DdsShapes.DdsObjectManagedBridge.2" tooltip="Customer" left="2456" top="5464">
                <control>
                  <ddsxmlobjstreaminitwrapper binary="00080000891300005d310000" />
                </control>
                <layoutobject>
                  <ddsxmlobj>
                    <property name="LogicalObject" value="dbo_DimCustomer" vartype="8" />
                  </ddsxmlobj>
                </layoutobject>
                <shape groupshapeid="0" groupnode="0" />
              </ddscontrol>
            </Value>
          </Annotation>
        </Annotations>
      </Dimension>
    </ObjectDefinition>
  </Create>

```

Connected. GFAA GFAA\Guy Adventure Works DW 200... 00:00:00

Figure 6–1. The metadata language of SSAS is an XML dialect called XMLA.

■ **Note** To generate the XMLA script for an SSAS object, you must use SSMS, not BIDS! In SSMS, connect to SSAS, right-click the object you want to script, and select Script Dimension as ► *Create to* ► New Query Editor Window.

I'm opening this discussion of cube data storage with the XMLA because you need to understand what Analysis Services considers to be data and what it considers to be metadata as you begin to plan your cube storage options.

The simplest way to think of this is as follows: Data is the content (rows) in the fact table. Metadata is everything else. By *everything else*, I mean cube definitions, dimension names, attribute names, hierarchy names, and, most important, dimension attribute values. The following list provides some examples from the samples available with SSAS to help explain this concept more fully:

- *Data*: All fact table rows from Fact Internet Sales, Fact Internet Sales Reason, and so on. Data does *not* include rows from the dimension tables; those rows are metadata and treated differently from data during dimension and cube processing.
- *Metadata*: Examples include names and all attributes for the Customer dimension called Customer, the Customer dimensional hierarchy Customer Geography, the Customer dimension attribute Marital Status, Customer attribute values Married, Single, or Unknown.

When you design storage for your cube, your primary concern is how to store the *data* (or the fact table rows, *not* the dimension table rows). This is because the largest amount of data in any cube is generally contained in the facts (or fact table rows). There can be exceptions to this in the case of huge dimensions. We will discuss this case in more detail later in this chapter.

We'll start our discussion of storage with data storage, and metadata storage will be covered later in this chapter. For data storage, you have three choices at the topmost level:

- *MOLAP*: Multidimensional OLAP stores a copy of the facts in your SSAS cube. This is not a one-for-one storage option. Owing to efficient storage mechanisms used for cubes, the resultant storage is approximately 10–20 percent the size of the original data; that is, if you have 1GB in your fact table, plan for around 200MB storage on SSAS. However efficient your system, when you chose MOLAP, be aware that you are choosing to make a copy of all source data.
- *HOLAP*: Hybrid OLAP does *not* make a copy of the facts in SSAS. It reads this information from the star schema source.
- *ROLAP*: Relational OLAP does *not* make a copy of the facts on SSAS. It reads this information from the star schema source.

Aggregations

The other major consideration is the quantity of and the storage of aggregations. An *aggregation* is a calculated, stored intersection of fact values. An aggregation is calculated at a level higher than the granularity of your fact table. In other words, if the grain of your fact table holds sales amounts for each product sold on each day by each employee, one possible aggregation would be to sum, and store, the daily facts at the week level. In this scenario, SSAS will use calculated aggregations to quickly return sales amounts for each product, for each employee, by year. If there are aggregations at the week level, SSAS

will use these as a starting point to calculate the needed yearly results and will not read each value from the day level.

An important difference between OLAP aggregations and materialized indexes on calculated columns in a relational database such as SQL Server is that the SSAS can use aggregations from *any* level in response to a query. The result is that full cube aggregation is *never* needed to optimize queries. In fact, *overaggregation* of OLAP cubes is a common design mistake. Baseline values for aggregation are defined a bit later in this section.

The storage type you select impacts where aggregations are stored. MOLAP, by default, creates no aggregations. Should you choose to add aggregations, they will be stored in the native SSAS format in your SSAS cube with the fact data. For HOLAP, only aggregations—not fact data—will be stored in your SSAS cube. For ROLAP, aggregations will be written back to the relational database. Figure 6–2 shows a conceptual rendering of a ROLAP aggregation table as part of a star schema. Note that the order of the column names reflect the positions in the hierarchy that are being aggregated and the type of aggregation performed.

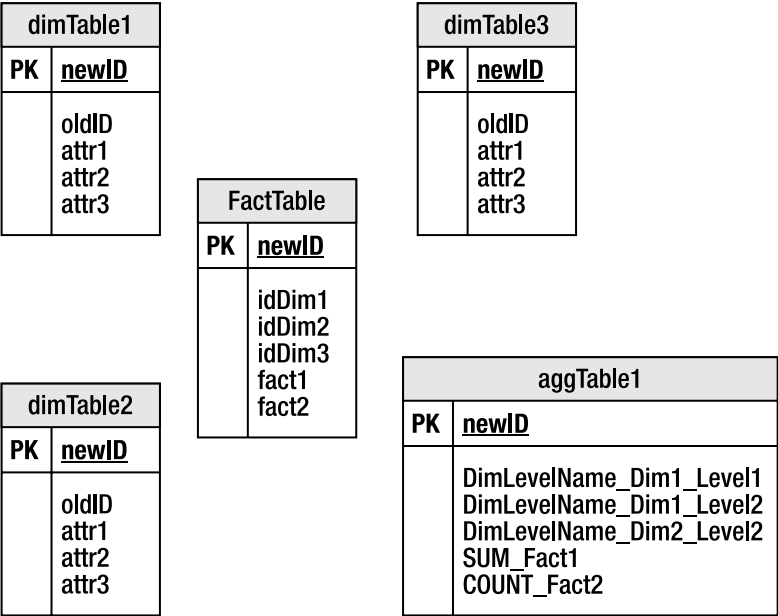


Figure 6–2. ROLAP storage creates and stores aggregations in the star schema source relational database. A side benefit of this storage is that these aggregations can be queried using T-SQL.

■ **Note** MOLAP is your only storage option if you are working with the Standard Edition of SSAS. If you are working with the Enterprise Edition of SSAS, you may choose MOLAP, HOLAP, or ROLAP storage.

MOLAP as Default in SSAS

So, which type of storage is best? Why is MOLAP, with 0 percent, aggregations the default mechanism? In our experience, MOLAP is most often chosen because it provides efficient storage and the fastest query results. Although the default level of aggregations is 0 percent, most of our customers choose to add at least some aggregations to their production cubes. The reason for doing this is that SSAS is optimized to quickly and easily add aggregations during cube and dimension processing, and the improvement in query results usually offsets the overhead in processing time and cube storage file sizes. We will drill into this topic a bit more deeply in the next section as well.

MOLAP is the storage default because the SSAS query engine is highly optimized for calculating aggregates on large quantities of data; the SSAS engine is more like Excel's calculation engine than that of SQL Server. Relational database engines, such as SQL Server, are designed to fetch subsets of data efficiently, not necessarily to perform the complex aggregations required for MOLAP. For many customers, MOLAP with 0 percent aggregations, or some small amount such as 20 percent, will produce queries that run quickly enough for all of their end users.

Adding Aggregations

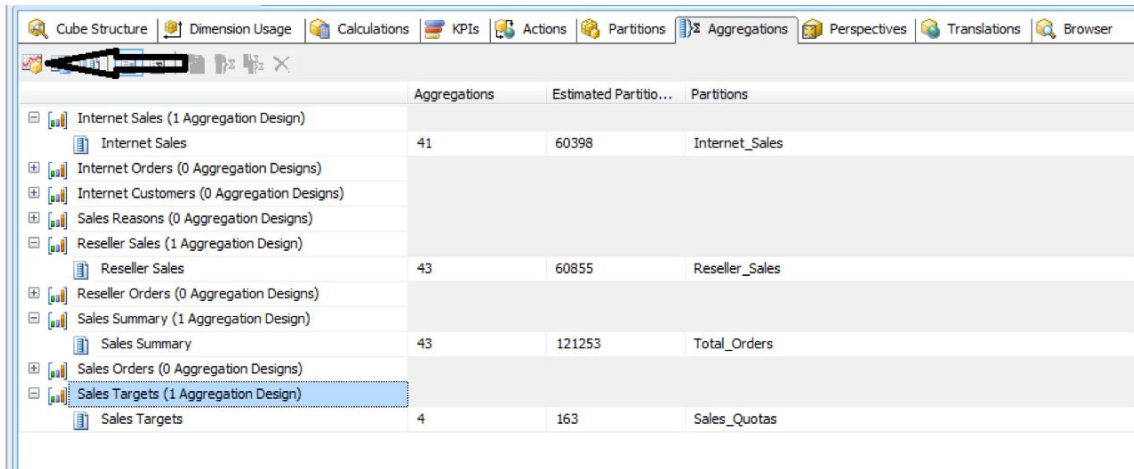
Why is it so important to add the correct amount of aggregations to your cube(s)? As stated previously, it may not be. It's important to remember that some SSAS cubes do not require aggregations to function acceptably. Similar to the idea that tiny OLTP tables need no relational indexes, if your cube is quite small (under 1GB) and if you have a small number of end users (100 or fewer), you probably won't have to add any aggregations at all.

Unlike adding indexes to a relational database, which can be time consuming if manually done, adding aggregations to an SSAS cube can be done pretty quickly via a couple of wizards (or tools) in BIDS. There are a couple of considerations when adding aggregations to a cube: aggregations increase cube processing times, and aggregations increase the storage space required for the cube on disk.

There are three tools that you can use in designing appropriate aggregations. Two—the Aggregation Design Wizard and the Usage-Based Optimization Wizard—are specifically for aggregation design. The third that we talk about is the SQL Server Profiler, which you can creatively apply as an aggregation design aid.

The Aggregation Design Wizard

The first tool is the Aggregation Design Wizard. You access the wizard by clicking the Aggregations tab in the Cube Designer in BIDS and then clicking the Design Aggregations button, highlighted in Figure 6-3.



| | Aggregations | Estimated Partitio... | Partitions |
|--|--------------|-----------------------|----------------|
| Internet Sales (1 Aggregation Design) | | | |
| Internet Sales | 41 | 60398 | Internet_Sales |
| Internet Orders (0 Aggregation Designs) | | | |
| Internet Customers (0 Aggregation Designs) | | | |
| Sales Reasons (0 Aggregation Designs) | | | |
| Reseller Sales (1 Aggregation Design) | | | |
| Reseller Sales | 43 | 60855 | Reseller_Sales |
| Reseller Orders (0 Aggregation Designs) | | | |
| Sales Summary (1 Aggregation Design) | | | |
| Sales Summary | 43 | 121253 | Total_Orders |
| Sales Orders (0 Aggregation Designs) | | | |
| Sales Targets (1 Aggregation Design) | | | |
| Sales Targets | 4 | 163 | Sales_Quotas |

Figure 6–3. The Aggregation Design Wizard will assist you with intelligent aggregation design for your cube.

In the first dialog box of the wizard, you review the assigned aggregation settings for your dimensions attributes. For each attribute, the wizard gives you four possible settings:

- *Default:* The aggregation designer chooses a default rule to apply. When using the default setting, key attributes will be used in all aggregations, and other attributes will be evaluated using the Unrestricted setting. We recommend using this setting.
- *Full:* Choosing this setting will include this attribute in all aggregations. We do not recommend using this setting, as it can make your cube extremely large.
- *None:* Choosing this setting will exclude this attribute from all aggregations.
- *Unrestricted:* Choosing this setting puts no restriction on the designer when evaluating an attribute.

Finally, you can click the Set All to Default button to set all aggregations to Default. Figure 6–4 shows the Review Aggregation Usage dialog.

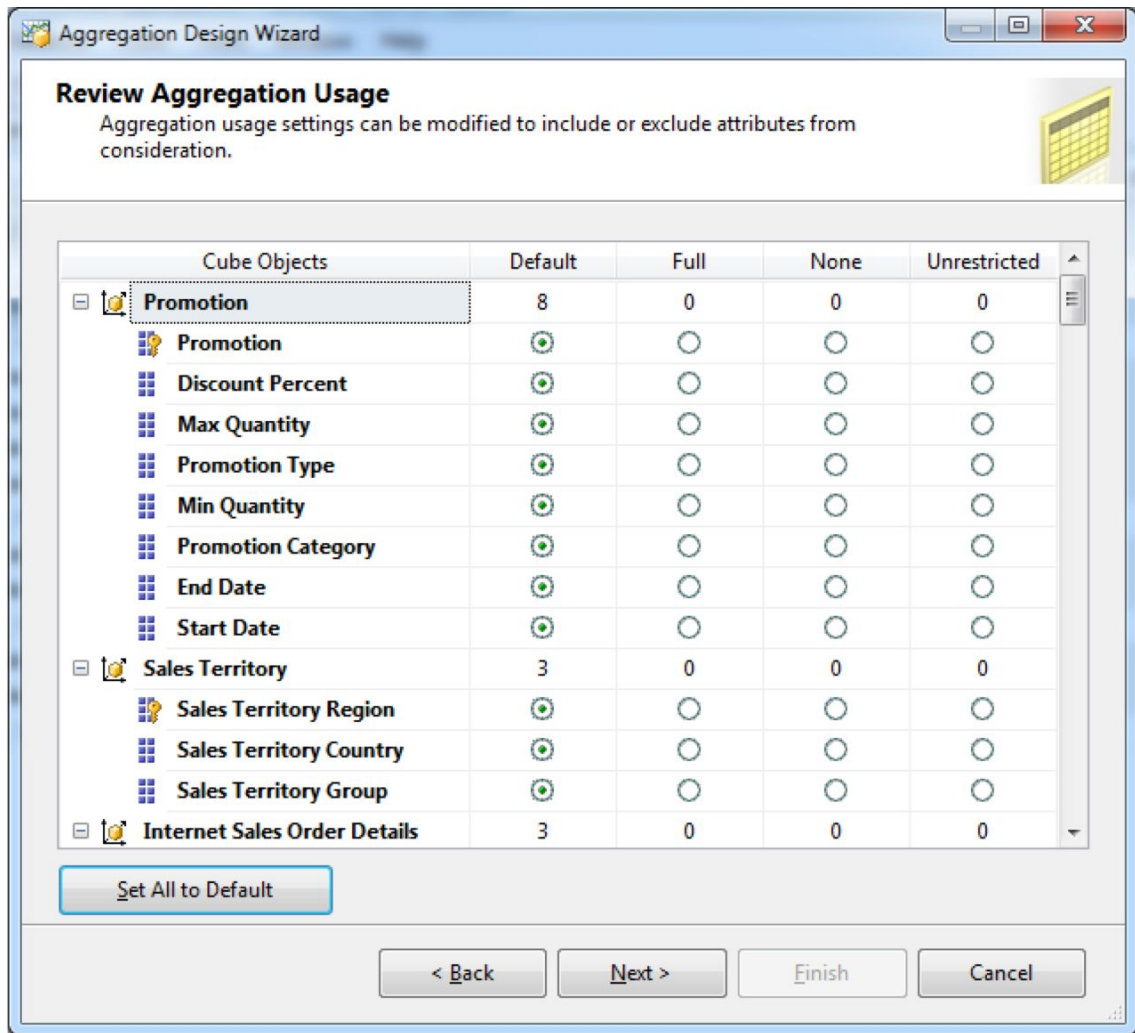


Figure 6–4. Use the Review Aggregation Usage dialog to define aggregation levels for your cube.

After finalizing your choices in the Review Aggregation Usage dialog, click Next to proceed to the Specify Object Counts dialog. In the Specify Object Counts dialog, you are asked to provide a count of the rows in the fact table, or you can ask SSAS to count the rows for you. Click Count to have SSAS obtain your record counts, and click Next. The Set Aggregation Options dialog will present you with four options for aggregation design:

- *Estimated storage reaches:* SSAS designs aggregations up to the size limit you input here (in megabytes or gigabytes) for storage of the resulting aggregations on disk.
- *Performance gain reaches:* SSAS designs aggregations up to this threshold that you input as a percentage increase in query performance speed.

- *I click stop*: The wizard stops adding aggregations when you click stop.
- *Do not design*: The wizard does not design any aggregations.

Select the “Performance gain reaches” option in the wizard, and enter 20% as your gain value. Click Start to begin processing. When processing completes, the Set Aggregation Options dialog will look like Figure 6–5.

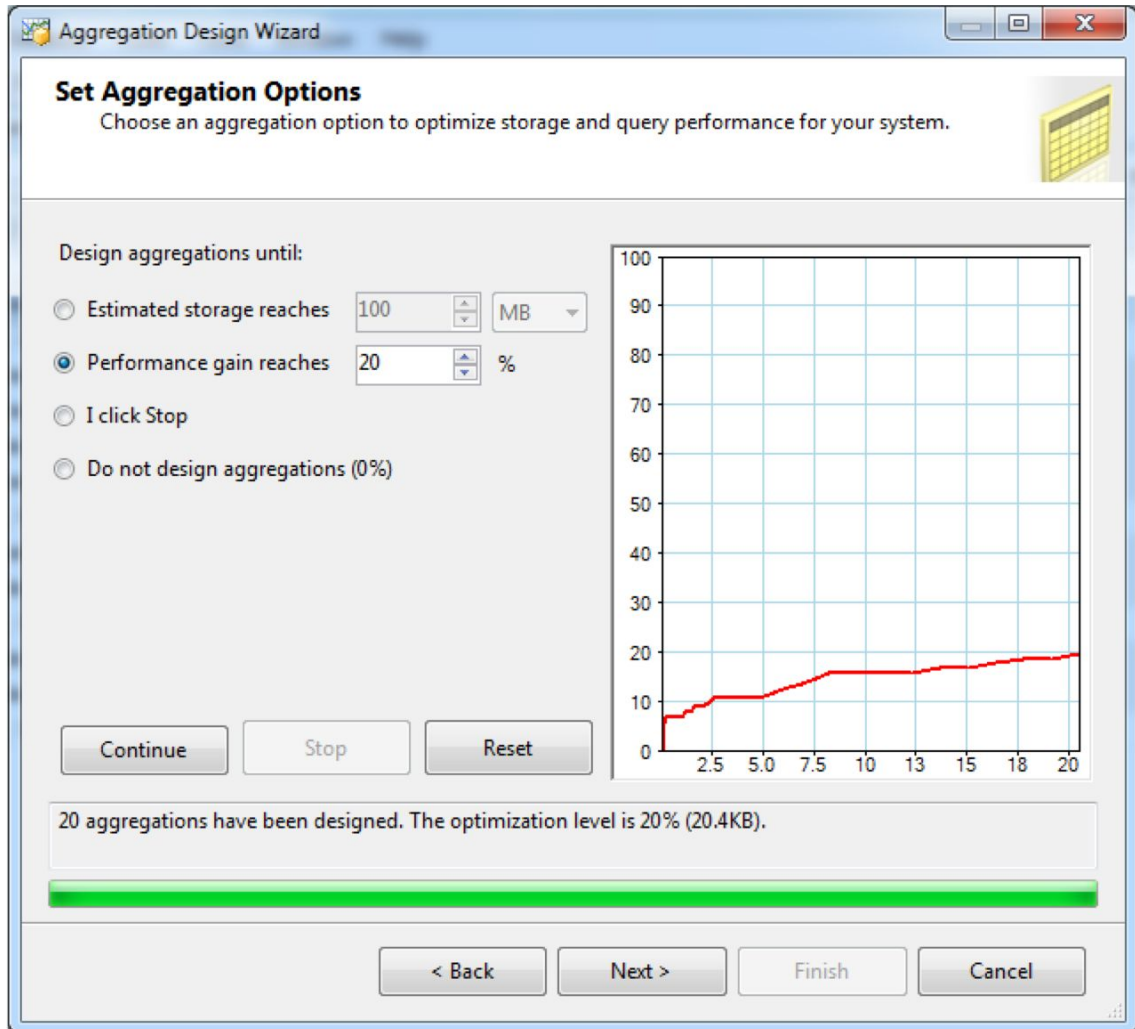


Figure 6–5. The Set Aggregation Options dialog, displaying the number of aggregations designed by the wizard.

Click Next; enter **AdventureWorksAggregations** in the Name input area, and click Finish.

As discussed earlier, there is no need to aggregate an SSAS cube 100 percent, because SSAS can use intermediate aggregations to answer queries. As a general rule of thumb, aggregating more than 50 percent is usually detrimental, as doing so increases cube-processing times and storage space without positively impacting query response times. Another consideration is that when the wizard completes its recommendations, you can choose whether to process the cube using those recommendations immediately, or to save the results for later processing.

The Usage-Based Optimization Wizard

The second tool you can use to more granularly define aggregations—the Usage-Based Optimization Wizard—is available by clicking the Usage Based Optimization button on the toolbar, next to the Design Aggregations button you used in the previous section. This wizard works by saving actual queries sent to the SSAS database. The saved queries are based on parameter values that you specify, such as start and end time, user name, and so on. The wizard then figures out which aggregations will best improve the performance of the queries that are run and which fall within the configured parameters. Because query performance is determined as much by the selected (and filtered) queries coming from your client applications as it is by the data. This makes effective use of the Usage-Based Optimization Wizard a very intelligent approach, because you are causing SSAS to create aggregations specifically for the particular queries, rather than just using the blanket approach of the Aggregation Design Wizard.

You must configure three SSAS properties prior to running the wizard. The first is `QueryLogConnectionString`. You set this value to the database connection string where you want to store the query log table. The data stored in this table will be retrieved by the Usage-Based Optimization Wizard (this process is similar to the usage of a trace table by the database tuning advisor for OLTP relational index optimization). To set the `QueryLogConnectionString` property, right-click the SSAS instance in SSMS, and then click Properties.

The second property is `CreateQueryLogTable`. Set this to True to have SSAS create a table that will log queries. This table will be used to provide queries to the wizard. This process is similar to using a trace table to provide queries to SQL Profiler for relational database query tuning. You can optionally change the default name of the query log table for the database you previously defined. This value is set to `OlapQueryLog` by default.

The third property to set is `QueryLogSampling`. The default is to only capture one out of ten queries. You will probably want to set this to 1 for your sampling purposes, so that every query within the defined parameter set is captured. Figure 6–6 shows the properties window for SSAS inside of SSMS where you can set all of these properties.

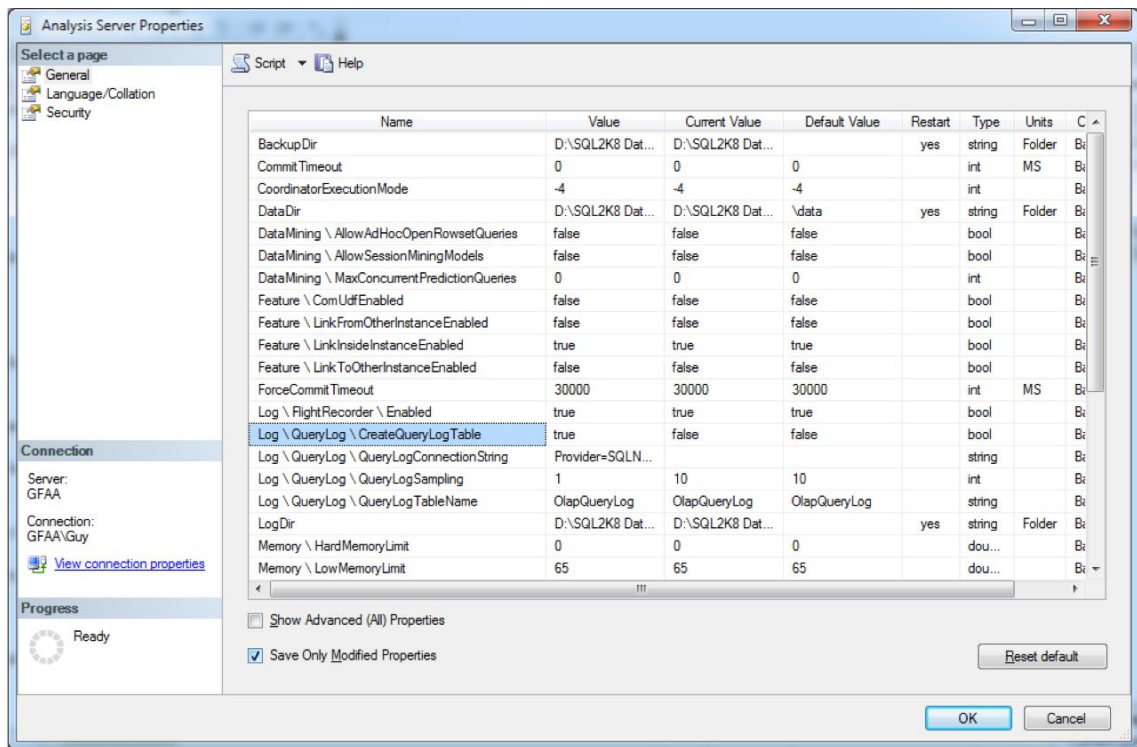


Figure 6–6. You must capture queries by setting the database connection string property for the `QueryLogConnectionString` value and setting the `CreateQueryLogTable` switch to `True` before you can use the *Usage-Based Optimization Wizard* in SSMS.

To run the wizard, click the Usage Based Optimization button. After you start the wizard, you ask SSAS to design aggregations based on any combination of the following parameter values: beginning date, ending date, specific user, and quantity of queries by percentage to total. Figure 6-7 shows a sample list of queries. In the wizard, you select the queries you want SSAS to design aggregations for.

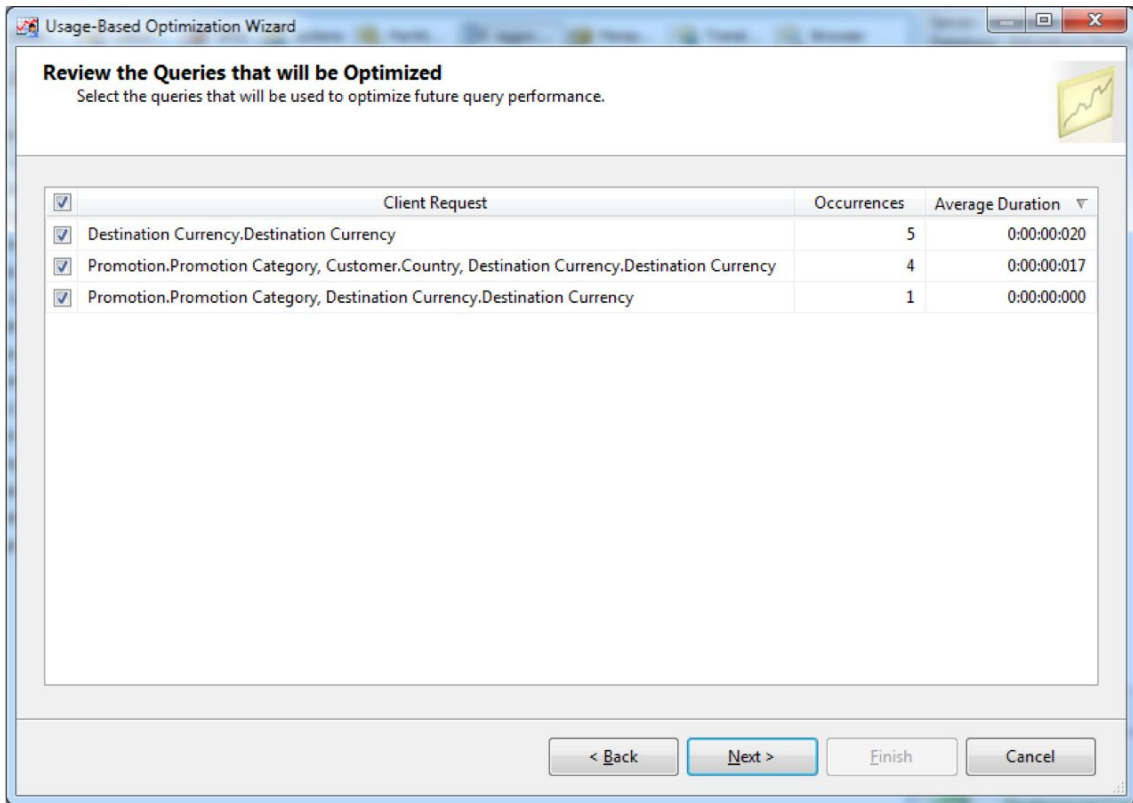


Figure 6–7. The Usage-Based Optimization Wizard allows you to select which queries you want SSAS to design aggregations for.

The SQL Server Profiler as an Aggregation Design Aid

Another interesting tool that you can use to design aggregations more intelligently is SQL Server Profiler. You may be familiar with SQL Server Profiler's capability to capture whatever traffic and information that you set in the trace definition for OTLP databases, but you may not be aware that you can use SQL Server Profiler to capture activity on Analysis Services. This allows you to easily see which MDX queries are taking up the most resources on the server.

One method of query tuning is to add aggregations, while another method of improving query performance is to rewrite the query in a more optimal way. Although this is possible, it is done much less often in the world of SSAS (than, for instance, rewriting T-SQL queries for SQL Server) because of the inherent complexity of MDX. Using SQL Server Profiler to identify those queries placing the heaviest load on the server will help you effectively target your tuning efforts, which more often involve adding aggregations than rewriting the MDX queries, for those queries that need it the most.

Chapter 13 is devoted to introducing MDX syntax. Figure 6–8 shows output from SQL Server Profiler while tracing SSAS activity.

| EventClass | EventSubclass | TextData | StartTime | CurrentTime | Duration |
|-----------------------|--------------------|---|-------------------------|--------------------------|----------|
| Progress Report End | 14 - Query | Finished reading data from the 'Internet_Sales_2006' partition. | 2010-10-24 22:40:156... | 2010-10-24 22:40:156.000 | 2 |
| Progress Report Begin | 14 - Query | Started reading data from the 'Internet_Sales_2007' partition. | 2010-10-24 22:40:156... | 2010-10-24 22:40:156.000 | |
| Progress Report End | 14 - Query | Finished reading data from the 'Internet_Sales_2006' partition. | 2010-10-24 22:40:156... | 2010-10-24 22:40:156.000 | 6 |
| Progress Report Begin | 14 - Query | Started reading data from the 'Internet_Sales_2008' partition. | 2010-10-24 22:40:156... | 2010-10-24 22:40:156.000 | |
| Progress Report End | 14 - Query | Finished reading data from the 'Internet_Sales_2007' partition. | 2010-10-24 22:40:156... | 2010-10-24 22:40:156.000 | 21 |
| Query Subcube | 2 - Non-cache ... | 00000100,000,000,0000000000000000000000,1111010000000000000000,00,100,0000000000000000... | 2010-10-24 22:40:156... | 2010-10-24 22:40:156.000 | 14 |
| Query Subcube | 2 - Non-cache ... | 00000100,000,000,0000000000000000000000,1111010000000000000000,00,100,0000000000000000... | 2010-10-24 22:40:156... | 2010-10-24 22:40:156.000 | 54 |
| Query Subcube | 1 - Cache data | 00000100,000,000,0000000000000000000000,1111010000000000000000,00,100,0000000000000000... | 2010-10-24 22:40:156... | 2010-10-24 22:40:156.000 | 22 |
| Query Subcube | 1 - Cache data | 00000000,000,000,0000000000000000000000,1111010000000000000000,00,100,0000000000000000... | 2010-10-24 22:40:156... | 2010-10-24 22:40:156.000 | 0 |
| Query End | 0 - MDXquery | SELECT NON EMPTY [(EF802185-1265-42FE-A0B7-0F14BA488034)Pivot7Axis1Set0] DIMENS... | 2010-10-24 22:40:156... | 2010-10-24 22:40:156.000 | 154 |
| Query Begin | 0 - MDXquery | DROP SET [Adventure Works].[EF802185-1265-42FE-A0B7-0F14BA488034]Pivot7Axis1Set3 | 2010-10-24 22:40:156... | 2010-10-24 22:40:156.000 | |
| Query End | 0 - MDXquery | DROP SET [Adventure Works].[EF802185-1265-42FE-A0B7-0F14BA488034]Pivot7Axis1Set3 | 2010-10-24 22:40:156... | 2010-10-24 22:40:156.000 | 1 |
| Discover Begin | 9 - MDXSchema_P... | <restriction base="urn:schemas-microsoft-com:xml-analysis" xmlns:soap="http://sc... | 2010-10-24 22:40:156... | 2010-10-24 22:40:156.000 | |
| Discover End | 9 - MDXSchema_P... | </restriction> | 2010-10-24 22:40:156... | 2010-10-24 22:40:156.000 | 3 |
| Query Begin | 0 - MDXquery | DROP SET [Adventure Works].[EF802185-1265-42FE-A0B7-0F14BA488034]Pivot6Axis1Set0 | 2010-10-24 22:40:158... | 2010-10-24 22:40:158.000 | |
| Query End | 0 - MDXquery | DROP SET [Adventure Works].[EF802185-1265-42FE-A0B7-0F14BA488034]Pivot6Axis1Set0 | 2010-10-24 22:40:158... | 2010-10-24 22:40:158.000 | 0 |
| Query Begin | 0 - MDXquery | DROP SET [Adventure Works].[EF802185-1265-42FE-A0B7-0F14BA488034]Pivot6Axis1Set3 | 2010-10-24 22:40:158... | 2010-10-24 22:40:158.000 | |
| Query End | 0 - MDXquery | DROP SET [Adventure Works].[EF802185-1265-42FE-A0B7-0F14BA488034]Pivot6Axis1Set3 | 2010-10-24 22:40:158... | 2010-10-24 22:40:158.000 | 0 |
| Query Begin | 0 - MDXquery | DROP SET [Adventure Works].[EF802185-1265-42FE-A0B7-0F14BA488034]Pivot5Axis1Set0 | 2010-10-24 22:40:158... | 2010-10-24 22:40:158.000 | |
| Query End | 0 - MDXquery | DROP SET [Adventure Works].[EF802185-1265-42FE-A0B7-0F14BA488034]Pivot5Axis1Set0 | 2010-10-24 22:40:158... | 2010-10-24 22:40:158.000 | 0 |


```

SELECT
  NON EMPTY [(EF802185-1265-42FE-A0B7-0F14BA488034)Pivot7Axis1Set0]
  DIMENSION PROPERTIES MEMBER_NAME, PARENT_UNIQUE_NAME ON COLUMNS,
  NON EMPTY [(EF802185-1265-42FE-A0B7-0F14BA488034)Pivot7Axis1Set4]
  DIMENSION PROPERTIES MEMBER_NAME, PARENT_UNIQUE_NAME ON ROWS,
  {
    [Measures].[Internet Sales Amount]
  }
  ON PAGES
FROM [Adventure Works]
CELL PROPERTIES VALUE, FORMATTED_VALUE, FORE_COLOR, BACK_COLOR

```

Figure 6–8. Profiler allows you to trace activities for SSAS. This can assist you with identifying long-running queries as candidates for aggregations.

Using Advanced Storage

In this section, we will look further into SSAS storage. These storage settings help you to balance query performance with data latency. In earlier sections, you learned that using MOLAP will provide the best query performance. Now, we will look into the other two storage settings, HOLAP and ROLAP. For both storage types, source data is not copied to your cube. While HOLAP aggregations are stored in SSAS, ROLAP aggregations are built and stored in your database.

Understanding ROLAP

ROLAP is often used for rapidly changing dimensions (RCDs), and huge dimensions (also called *monster dimensions*). As an example, let's say that you are modeling a dimension that contains employee information for a fast-food restaurant chain. The chain has very high employee turnover, as is typical in the fast-food industry. However, a business requirement is to be able to retrieve the most current employee name from the employee dimension at all times with no latency. This type of requirement may lead you to choose a ROLAP dimension.

Another example is if you have a huge number of members in a dimension. A business example is that you are modeling the customer dimension for an international shipping company. It is a business requirement to have the name of every customer for all time included in your cube. This may mean that you must include millions, or eventually even billions, of customer names in the dimension. The storage limits for SQL Server tables (maximum number of rows) is still much larger than those in SSAS dimensions.

Despite the fact that you may have business situations that warrant the consideration of ROLAP dimensions, you should test to make sure that your infrastructure will provide adequate performance given the anticipated load. If you are considering ROLAP dimensions, be sure to test with production level of data before you deploy this configuration into a production environment.

Understanding HOLAP

In Hybrid OLAP (HOLAP), your aggregations are stored in SSAS, while your source data remains in your RDMBS. HOLAP takes advantage of MOLAP query performance, lowers cube processing times, and reduces cube size. HOLAP storage has lower latency than MOLAP. However, queries needing to drill through to source data will perform poorer than MOLAP queries, since this detail data is not in your cube. Also, HOLAP storage requires the largest percentage of aggregations. If a good deal of your analysis is based on aggregations only, HOLAP may just be worth a try.

Considering Non-MOLAP Storage

Storing cube data and metadata in MOLAP results in the best query performance, so why use any other type of storage? Most typically, you will only be interested in these options if your cube is relatively large, 250GB or greater, for example, or if storage space is problematic. Another reason to consider ROLAP or HOLAP storage is if cube-processing times are excessive for your particular business needs. Also huge dimensions sometimes warrant the use of ROLAP storage—*huge* being defined as those dimensions containing millions of members. A final consideration for nondefault storage is the business case, which requires real-time information. Be diligent in understanding requirements for this last situation. Using *Proactive Caching* (described more fully in the upcoming section of the same title) allows for near real-time results—latency of seconds in some cases—without the performance hit that pure ROLAP causes. Most prefer the improved query performance of MOLAP and are willing to trade a bit of latency to get that level of performance.

Although you can change the storage type for an entire cube, it is more common to change the physical storage design for a portion of the cube. In SSAS, a portion of a cube is called a *partition*. Creating partitions in a cube requires the Enterprise Edition of SSAS. A SSAS partition is a logical division of a cube. A partition can be based on a particular fact table that is part of a cube, or it can be based on a portion of a fact table. For example, a fact table could be partitioned by month. In the SSAS interface, the terms *partition* and *measure group* are often used interchangeably. This is a bit of a misnomer because even cubes with a single (default) partition can have more than one measure group associated with that partition. In other words, there is not necessarily a one-to-one ratio between a measure group and a partition.

To create a partition in BIDS, you click the partition tab on the cube design surface, and then click the *New Partition...* link on the design surface as shown in Figure 6–9. You are asked to select which Measure Group and Partition (Fact Table) Source you want to partition in the wizard. The Partition Source is a DSV in the SSAS database. Alternatively, you can enter a table name in the “Filter tables” text box and click the Find Tables button to restrict the source to a particular source table. This dialog box is shown in Figure 6–10.

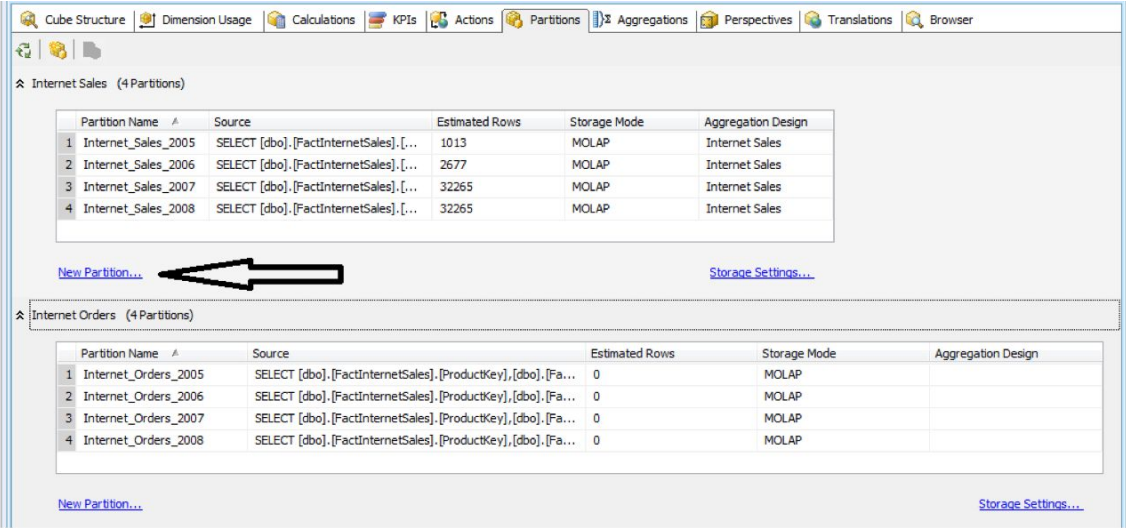


Figure 6–9. To vary the cube storage by a subset of the cube, you must first create partitions in BIDS.

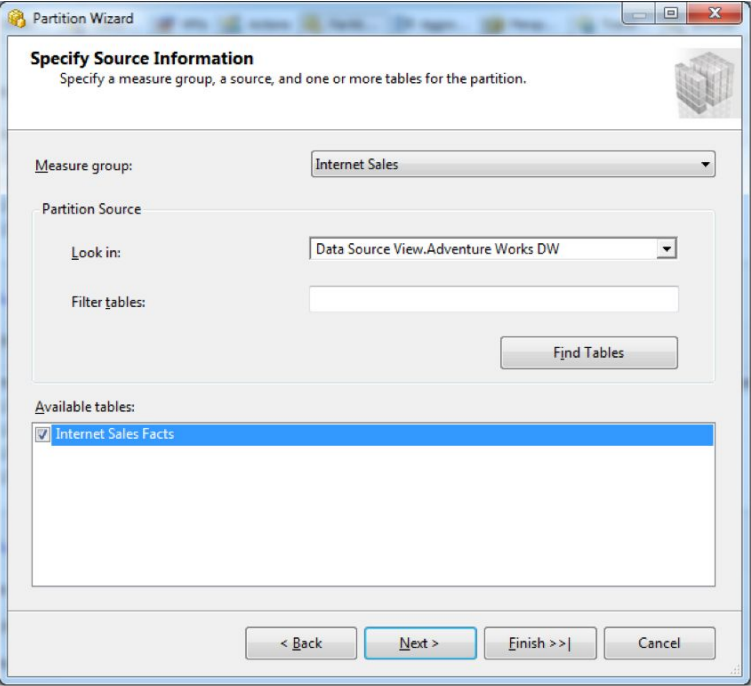


Figure 6–10. The Partition Wizard asks for information about which measure group you want to partition.

Whether you want to implement some type of nondefault storage for the entire cube or for one of the partitions in a cube, the procedure is identical. In the BIDS cube designer, you click the Storage Settings link on the Partitions tab. This displays a wizard that you can adjust either by sliding the slider bar or by setting custom storage options. Figure 6–11 shows the default setting, MOLAP.

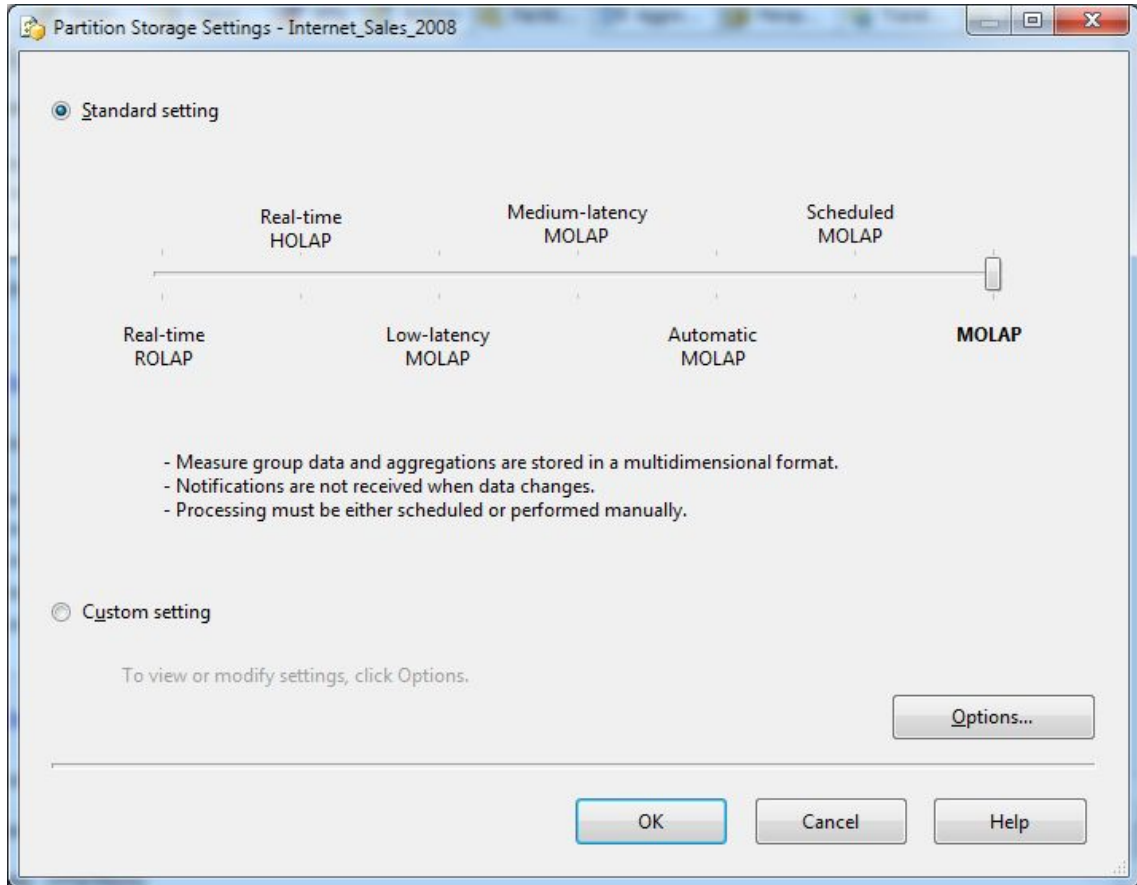


Figure 6–11. To adjust the default storage setting of MOLAP for a cube or for a partition (which is associated with a particular measure group in SSAS), you access the Measure Group Storage Settings dialog box using the Partitions tab of the cube browser in BIDS.

Although the slider provides a brief explanation of the other settings, you probably need a fuller explanation to effectively select something other than the default. Note that a new feature called *Proactive Caching* is enabled for all storage modes other than the default MOLAP mode. Proactive Caching is covered in the next section of this chapter. Here's a list of the impact of changing the default setting in the Storage Settings dialog box.

- *MOLAP (default)*: Source *data* is copied from the star schema fact tables to SSAS as MOLAP data; source *metadata* (which includes cube and dimension structure and dimension data) and *aggregations* are copied (for dimension data) or generated (for all other metadata and aggregations) and then are stored in MOLAP format on SSAS. Proactive caching is not used.
- *MOLAP (nondefault)*: Source *data* is copied; *metadata* and *aggregations* are stored in MOLAP format on SSAS, and proactive caching is enabled. This includes Scheduled-, Automatic-, Medium-, and Low-latency MOLAP.
- *HOLAP*: Source *data* is *not* copied; *metadata* and *aggregations* are stored in MOLAP format in SSAS. Proactive caching is enabled.
- *ROLAP*: For a cube, source *data* is *not* copied; *metadata* is stored in MOLAP format on SSAS, and *aggregations* are stored in the star schema database. For a dimension, *metadata* is *not* copied but is simply read from the star schema database table(s), and proactive caching is enabled.

Handling Huge Dimensions

Like so many of the advanced storage features, ROLAP dimensions require the Enterprise Edition of Analysis Services. You would use ROLAP typically only for dimensions with millions of members; an example might be the customer dimension. This means that the dimensional attribute values will not be copied to and stored on SSAS; rather, they will be retrieved directly from the relational source table or tables. To set a dimension as a ROLAP dimension, go to the Properties window for that dimension, and change the *StorageMode* property from the default MOLAP to ROLAP, as shown in Figure 6–12.

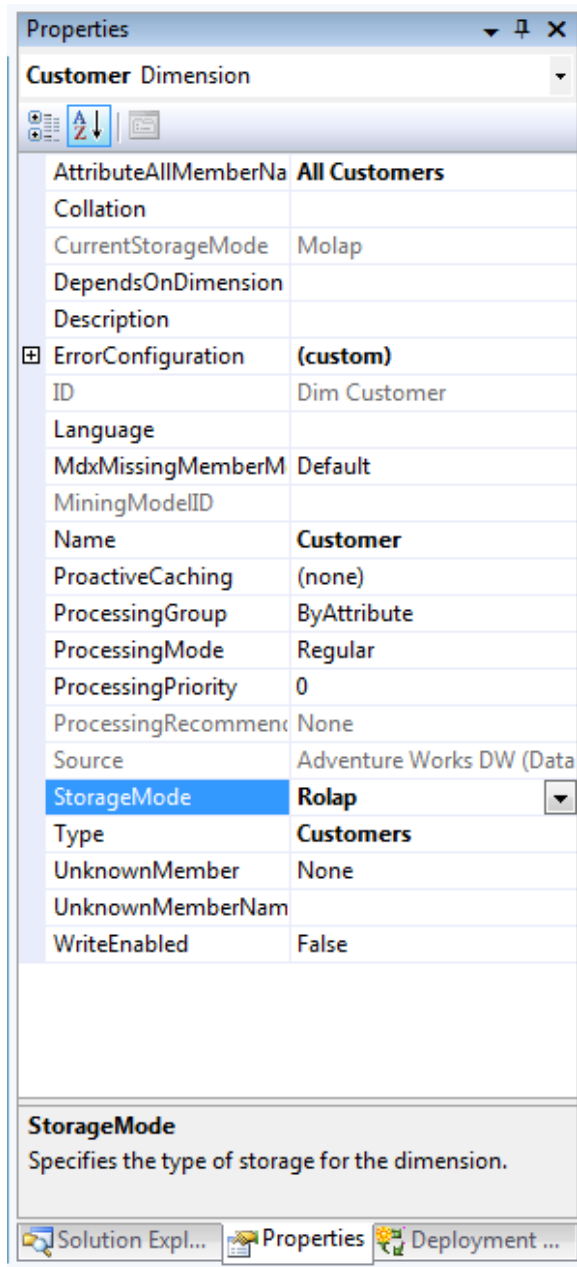


Figure 6–12. You can change the default dimension storage mode from MOLAP to ROLAP if your business scenario warrants it. This is usually done for very large dimensions, that is, millions of members.

Summarizing OLAP Storage Options

If you are working with the Standard Edition of SSAS, the only consideration for you is whether or not to add aggregations to your cube because non-MOLAP storage and cube partitions are enterprise-only features. In our experience, for cubes of 100GB or less, it is often prudent to add aggregations to the 20 to 30 percent query-improvement level. If particular queries are problematic, you can use the Usage-Based Optimization Wizard to capture them and have SSAS add aggregations specifically for those queries. The procedure for this was described in the “Adding Aggregations” section earlier in this chapter.

If you are working with the Enterprise Edition of SSAS, you have several choices in terms of cube storage and aggregation:

- *Should I use partitions?* If your cubes are 250GB or larger, you’ll probably want to partition them by either months, quarters, or years (assuming your time granularity is by the day and not the hour or minute).
- *Should I add aggregations?* If you are considering partitioning your cubes because they are large, you should also consider customizing the amount of aggregations based on the size of partition and frequency of query. In addition to the tools reviewed previously, you may also choose to use SQL Profiler to track activity on your cubes at a more granular level so that you can aggregate appropriately.
- *Should I use MOLAP, ROLAP, or HOLAP?* Generally, you’ll use something other than MOLAP only if you choose to use partitions. Typically, current data (partitioned on some time value, for example, months) is stored as MOLAP with a relatively high level of aggregation (25–30 percent), with older data partitions stored as HOLAP or ROLAP with a lower level of aggregation.

Table 6–1 summarizes storage information.

Table 6–1. Storage Options for SSAS Cubes

| Type | Data (Facts) | Metadata | Aggregations | Reason |
|--------------------|---------------|------------|----------------|-----------------------------|
| MOLAP cube* | SSAS (copied) | SSAS | SSAS | Default/fastest query** |
| HOLAP*** cube | Not copied | SSAS | SSAS | Good for archive partitions |
| ROLAP*** cube | Not copied | SSAS | Written to SQL | Good for storage issues |
| ROLAP*** dimension | Not copied | Not copied | Written to SQL | Real-time dimensions |

* Cube or cube partition

** SSAS Standard Edition supports only MOLAP storage with the default number of partitions per cube (1).

*** Requires the Enterprise Edition of SSAS

If you are using the Enterprise Edition of SSAS, you have one additional storage configuration to consider: Proactive Caching.

Using Proactive Caching

Proactive caching enables your end users to access your BI data with all the speed and power of SSAS but without the typical latency (often one business day) between the OLTP source and OLAP data. Configuring proactive caching is the method by which you manage the MOLAP cache.

The *MOLAP cache* is an in-memory storage location created automatically by SSAS. The cache includes fact and dimension data and occasionally aggregations. This information is placed in the cache area after MDX queries are executed against the SSAS cubes. Figure 6–13 shows a conceptual rendering of the MOLAP cache.

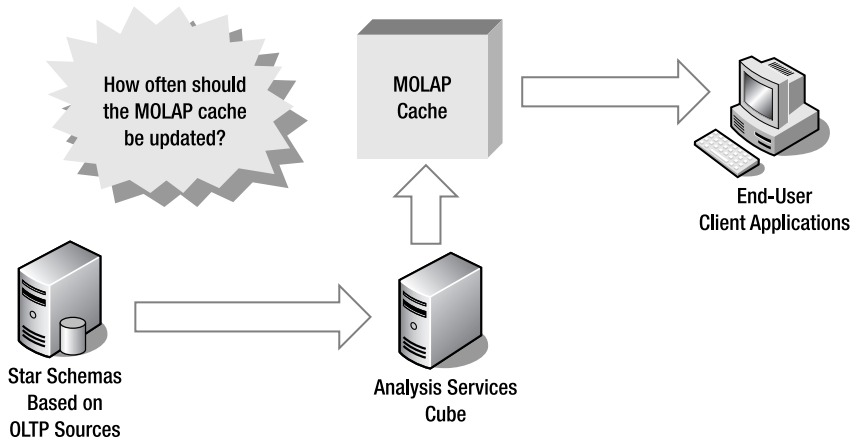


Figure 6–13. Proactive caching settings allow you to manage the update/rebuild frequency of the MOLAP cache for your cube or partition(s).

Note that proactive caching happens in near (but not exactly) real time. The nearer to real time, the more overhead is added to the system. That's why SSAS has six different options for you to choose from when configuring proactive caching using the measure group processing tool. You can achieve still more finely grained control using the Custom Options dialog box in this section and even more control by configuring the property section for any particular measure group. Also, remember that you can only configure storage and caching settings at the level of the partition, or subset of the measure group or fact table, if you have already set up partitions in your solution.

■ **Note** Proactive caching is *not* for every BI solution. Using it effectively requires you to either read your OLTP data directly as the source for your cube, or you read a cached copy of your data. Another option, if your source data is stored in SQL Server, is to read your OLTP data using Snapshot isolation level. To use these options, your source data must be very clean. If you need to do cleansing, validation, and/or consolidation during ETL processing, proactive caching is not the best choice for your solution.

Let's start with a more complete explanation of the choices available in the Measure Group Storage Settings dialog box, as they relate to proactive caching settings. The first choice you'll make is whether to use MOLAP, HOLAP, or ROLAP data storage for your cube. In most cases, due to the superior query performance, you'll select some version of MOLAP. The proactive caching configuration choices for MOLAP are as follows:

- *Scheduled MOLAP*: In this setting, the MOLAP cache is updated per a schedule (whether the source data changes or not); the default is once daily. This sets the rebuild interval to one day.
- *Automatic MOLAP*: In this setting, the cache is updated whenever the source data changes. It configures the silence interval to 0 seconds and sets no silence override interval.
- *Medium-latency MOLAP*: In this setting, the outdated caches are dropped periodically (the default is a latency period of 4 hours). The cache is updated when data changes (defaults are a 10-second silence interval and a 10-minute silence override interval).
- *Low-latency MOLAP*: In this setting, outdated caches are dropped periodically (the default is a latency period of 30 minutes). The cache is updated when data changes (defaults are a 10-second silence interval and a 10-minute silence override interval).

■ **Tip** To understand the “silence interval” property, think of this question: how long should the cache wait to refresh itself if there are no changes to the source data? To understand the “silence override interval” property think of this question: what is the maximum amount of time after a notification (of source data being updated) is received that the cache should wait to start rebuilding itself?

If you select HOLAP or ROLAP, the proactive caching settings are as follows:

- *HOLAP*: In this setting, outdated caches are dropped immediately (configures the latency period to 0 seconds). The cache is updated when data changes (defaults are a silence interval of 0 seconds and no silence override interval).

- **ROLAP:** In this setting, the cube is always in ROLAP mode, and all updates to the source data are immediately reflected in the query results. The latency period is set to 0 seconds.

Selecting the Options button on this same property sheet allows you to manually adjust the cache settings, options, and notification values, as shown in Figure 6–14.

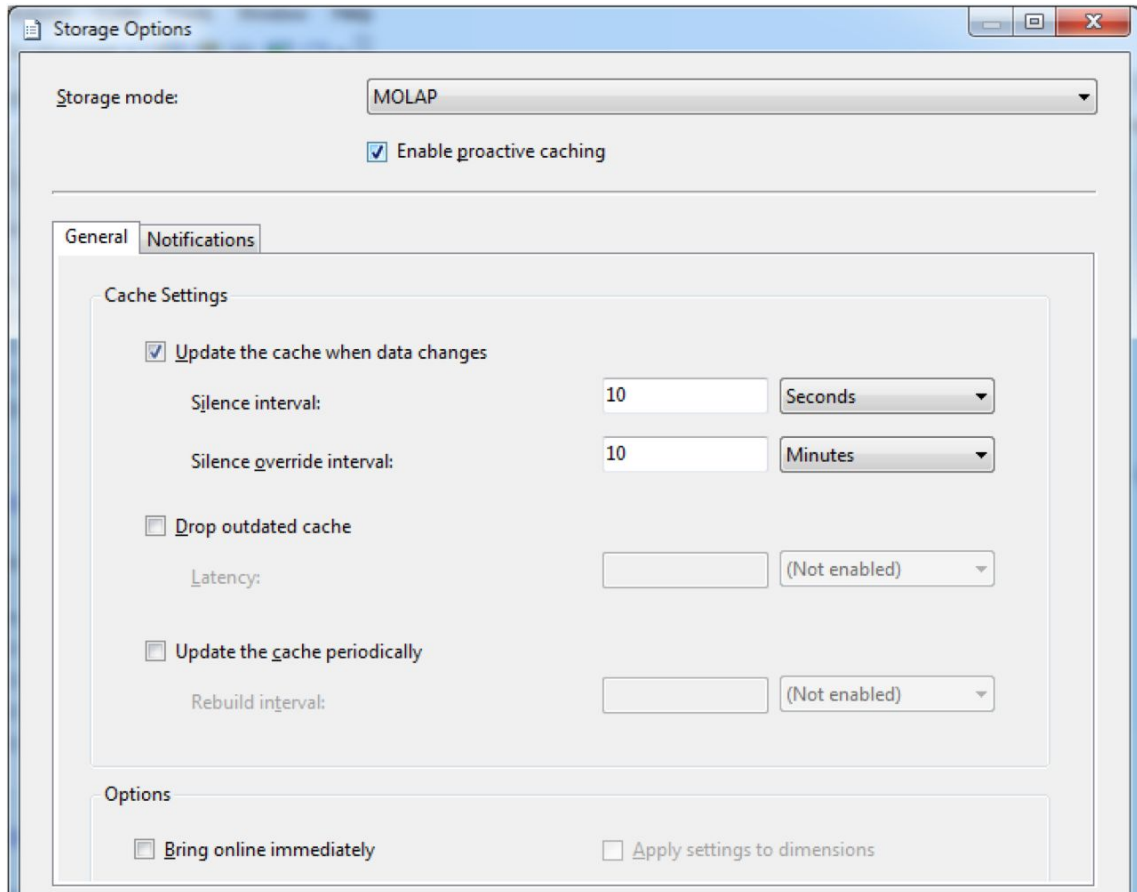


Figure 6–14. The custom settings for proactive caching include the General and Notifications tabs of the Storage Settings dialog box in the Partitions tab of BIDS. Here you can more granularly adjust the cache and notification settings.

Fine-Tuning Proactive Caching

You can fine-tune proactive caching settings for a particular measure group. Managing your proactive cache settings allows you to specify cache properties, such as how often to rebuild the cache and how often to drop an outdated cache. Table 7-2 provides a summary of the settings available.

Table 7-2. Proactive Caching Settings

| Setting | Choices | Description |
|-------------------------|----------------------------|--|
| Aggregation Storage | Regular; MOLAP only | Applies to partitions only |
| Enabled | True; False | Turns proactive caching on or off |
| ForceRebuildInterval | Time value | Maximum time increment to rebuild the cache whether the source data has changed or not; default is -1, which equals infinity |
| Latency | Time value | Maximum time to wait to rebuild cube; default is -1, which equals infinity |
| OnlineMode | Immediate; OnCacheComplete | Tells whether new cache will be available immediately or only after it's been completely rebuilt |
| SilenceInterval | Time value | Maximum time the source data has no transactions before the cache is rebuilt; default is -1, which equals infinity |
| SilenceOverrideInterval | Time value | Maximum time to wait after a data change notification in the source data to rebuild the cache; overrides SilenceInterval value; default is -1, which equals infinity |

Setting Notifications for Proactive Caching

You can adjust the notification settings, regarding data changes in the base OLTP store, by using the Notifications tab of the Storage Options dialog box. Notification Services allows clients to subscribe to interesting events and receive notification of changes. There are three types of notifications possible using this dialog box:

- *SQL Server:* For this option, you'll need to specify tracking tables in the relational source database. This option uses trace events and requires that the service account for SSAS has database owner (dbo) permissions on the SQL database that contains the tracking table.
- *Client Initiated:* As for SQL Server, you'll need to specify tracking tables in the relational source database. This is used when notification of changes will be sent from a client application to SSAS.

- *Scheduled Polling*: For this option, you'll need to specify the polling interval time value and whether you want to enable incremental updates and add at least one polling query. Each polling query or queries is also associated with a particular tracking table.

■ **Tip** Notification Services includes an adaptor for SSAS. Using this adapter, a custom client application could subscribe to an interesting event by specifying a custom MDX query (which would include the data to be inspected, the frequency of polling, and the notification scheme). For example, a plant manager could request notification if the number of defects per hour on an assembly line had risen above a specified level.

Deciding Between OLTP and OLAP Partitioning

If you are working with the Enterprise Edition of SQL Server, you now have the ability to do relational table partitioning in your star schema source tables. This strategy can complement cube partitioning you implement using SSAS, or you can choose to partition only on the relational side. The consideration then becomes which type of partitioning is appropriate for your BI solution?

Relational Table Partitioning in SQL Server

Table partitioning is the ability to position data from the same table on in different filegroups, often in different physical locations (disks), while that data appears to continue to originate from the same logical table from the end-user's perspective. This simplifies management of very large databases (VLDBs), and in particular, management of very large tables. The large tables that we are concerned about here are, of course, fact tables.

Fact tables commonly contain millions or tens of millions of rows. In fact, support for these huge fact tables is one of the reasons that the BIGINT datatype, which can house over four billion rows, was introduced in SQL Server 2000. Relational table partitioning can simplify administrative tasks and general management of these, often large or even huge, data sources. An example of this is that backups can be performed much more efficiently on table partitions, rather than on entire fact tables.

Although relational table partitioning is relatively simple, several steps are involved in implementing it. Here's a conceptual overview of the technique:

1. Identify the tables that are the best candidates for partitioning. For OLAP projects, as mentioned, these will generally be the fact tables.
2. Identify the value (or column) to be used for partitioning, usually a date field; a unique constraint must be implemented on this column of the tables that will participate in partitioning.
3. Implement the physical architecture needed to support partitioning, that is, install the physical disks.
4. Create filegroups in the database for each of the new physical disks or arrays.
5. Create secondary database files (.ndf files), for the SQL Server database where the tables to be partitioned are contained, and associate these .ndf files with the filegroups you created in Step 4.

6. Create a partition function. This creates the buckets to distribute the sections of the table into. This is often done by date range, that is, from *xxx* to *yyy* date (most often monthly or annually).
7. Create a partition scheme. This associates the buckets you created previously to a list of filegroups, in which there is one filegroup for each time period (that is, each month or year).
8. Create the table, usually the fact table, based on the partition scheme that you created earlier. This splits the table into the buckets you've created.

OLAP Partition Configurations

One other consideration in the world of partitions returns us to SSAS cube partitions. With the Enterprise Edition of SSAS, it is possible to define cube partitions as local (the default) or remote. A *local* partition is one in which the data is stored in the same database as the definition. A *remote* partition is one in which the data is stored in a database separate from that containing the definition of the partition.

The primary reason to consider using remote partitions is to do a kind of load balancing in the SSAS environment. You would use remote partitions to implement load-balancing situations where your primary SSAS server was stressed due to (usually) a large number of users executing complex queries. By using remote partitions, you can split the processing work across multiple physical servers. Additional considerations with remote partitions include the following:

- *Remote partitions*: These partitions can use MOLAP, HOLAP, or ROLAP storage. They can also use proactive caching. Remote partitions store information on the remote server.
- *Remote MOLAP*: Data and aggregations for the remote partition are stored on the remote server.
- *Remote HOLAP*: Aggregations for the remote partition are stored on the remote server; data is read from the OLTP source.
- *Remote ROLAP*: Nothing is stored on the remote server; both data and aggregations are read from the OLAP source.

Choosing Cube and Dimension Processing Options

Now that we've covered storage, aggregations, partitions, and caching, we are ready to review cube and dimensions processing option types. Dimensions must be completely and correctly processed either prior to or at the beginning of a cube process. The best way to understand this is to remember that dimensional data is the metadata or the structure of the cube itself, so the metadata must be available before the data can be loaded into the cube.

During development, you will most often do a full process on your cube whenever you need to view the results of a change that you've made. This option completely erases and rebuilds all data and metadata. For some customers, this simple method of updating the cube can be used in production as well. A *full process* is, of course, a complete overwrite on rebuild. This is only practical for the very smallest cubes—a couple of gigabytes in size maximum.

In the majority of cases, you will choose the more granular processing options after moving your cube to a production environment. The first consideration is the ability to separate processing of

dimensions from the cube. In this chapter, we'll review the steps for processing using BIDS. In Chapter 8, we'll discuss automating these cube and dimension refreshes using SSIS tasks and workflows.

To process a cube, right-click the cube name in the Solution Explorer in BIDS, and select Process. You'll see the dialog box shown in Figure 6–15.

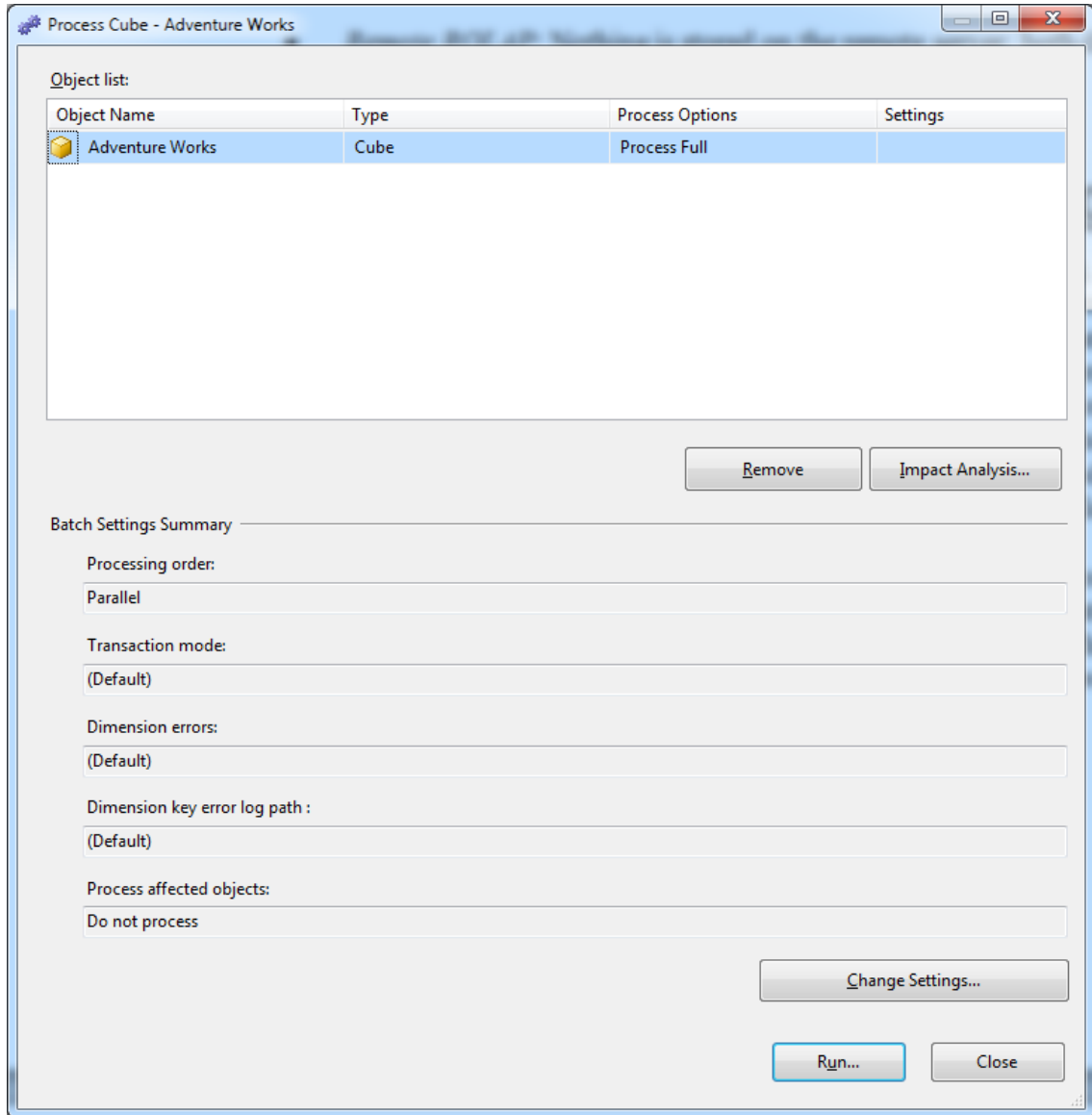


Figure 6–15. To process a cube, right-click the cube name in BIDS, click Process, and select the process type and options in the dialog box.

The following is a more complete explanation of the available selections for process options for both cubes and dimensions than what's offered in the dialog. Some options are only available for cubes or for dimensions as noted in the list:

- *Process Default*: SSAS detects the current state of the cube or dimension and then does whatever type of processing (full or incremental) is needed to return the cube or dimension to a completely processed state.
- *Process Full*: SSAS completely reprocesses the cube or dimension. For a cube, this will include all the objects contained within it, for example, dimensions. Full process is required when a structural change has been made to a cube or dimension. For a dimension, one example would be when an attribute hierarchy is added, deleted, or renamed.
- *Process Data*: SSAS processes data only and does not build any aggregations or indexes. SSAS indexes are not the same as relational indexes; SSAS indexes are generated and used by SSAS internally during the aggregation process.
- *Unprocess*: SSAS drops the data in the cube or dimension. If there are any lower-level dependent objects, for example, dimensions in a cube, those objects are dropped as well. This option is often used during the development phase of a BI project to quickly clear out erroneous results.
- *Process Index*: SSAS creates or rebuilds indexes for all processed partitions. This option results in no operation on unprocessed objects.
- *Process Structure (cubes only)*: SSAS processes the cubes and any contained dimensions but does not process any mining models.
- *Process Incremental (cubes only)*: SSAS adds newly available fact data and processes only the affected partitions. This is the most common option used in day-to-day production.
- *Process Update (dimensions only)*: SSAS forces an update of dimension attribute values. This value is to dimension processing as the incremental value is to cube processing; that is, Process Update adds new members for dimensions just as Process Incremental adds new facts for cubes.

■ **Note** Aggregation processing behavior in dimensions depends on the `AttributeRelationship RelationshipType` property. If this property is set to the default value (`Rigid`), aggregations are incrementally updated on incremental process of the cube or update of the dimension. If it is set to the optional (or nondefault) value (`Flexible`), aggregations are fully reprocessed for cube/dimension incremental updates. Also, if you set the dimension processing mode for a dimension to `lazyAggregations`, flexible aggregations are reprocessed as a background task, and end users can browse the cube while this processing is occurring.

An optimization step you can take to reduce processing times for your dimensions is to turn off the `AttributeHierarchyOptimized` property for dimensional attributes that are only viewed infrequently by end users. To adjust the `AttributeHierarchyOptimizedState` property, you'll open the Properties dialog box for the particular `DimensionalAttribute` and set the property value to `NotOptimized`. Figure 6-16

shows the property sheet for the Education attribute in the Customer dimension. Setting the value to `NotOptimized` causes SSAS to not create supplementary indexes (usually created by default) for this particular attribute on dimension or cube process. This can result in slower query times, so change this setting only for rarely browsed attributes.

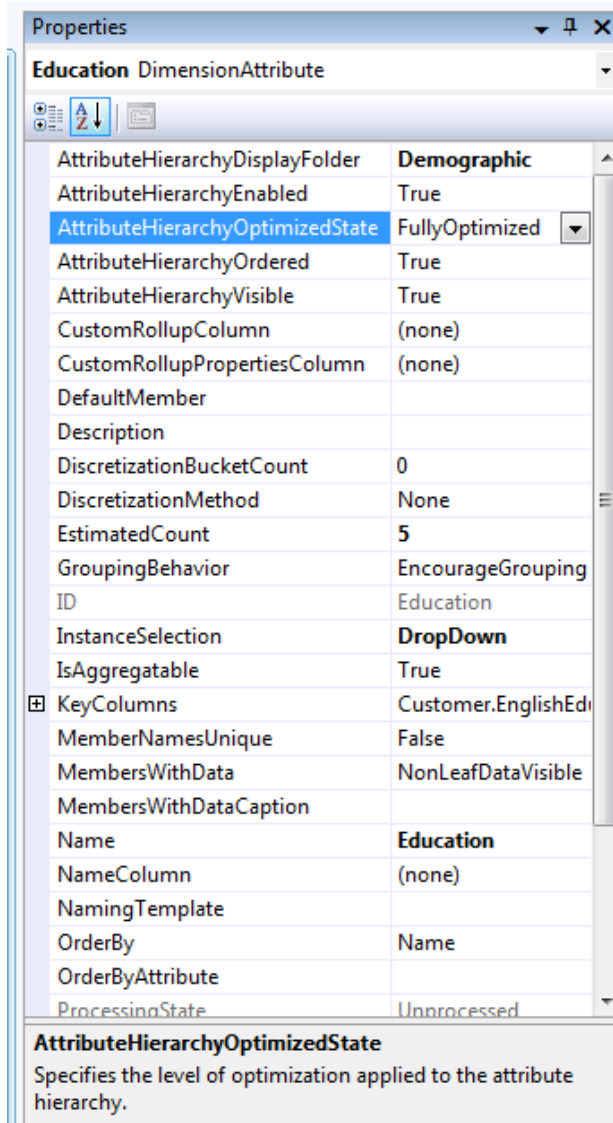


Figure 6–16. You can improve dimension processing times by adjusting the `AttributeHierarchyOptimizedState` property of rarely browsed DimensionalAttributes.

Tip To identify infrequently queried attributes, you could either use a capture of queries from SQL Server Profiler or read the content of the LogTable after running the Query Optimization Wizard.

The final consideration when processing cubes and dimensions is whether you'll need to adjust any of the processing options. You access these options using the Change Settings button in the Process Cube dialog box. Here, you have two tabs to work with. In the first tab, Processing Options, you can set the following values:

- *Parallel or Sequential processing*: If parallel, maximum number of parallel tasks
- *One or separate transactions*: For sequential processing
- *Writeback table*: Use existing or create new
- *Process dependent objects*: Off or on

The second tab, Dimension Key Errors, allows you to configure the behavior of errors during processing. You can either use the default error configuration or set a custom error configuration. When using a custom error configuration, you can specify the Key Error Action, the Processing Error Limit, the behavior to result for reaching Specific Error Conditions (Key Not Found, Duplicate Key, Null Key Converted to Unknown, or Null Key Not Allowed), and the Error Log Path location. Figure 6-17 displays a sample.

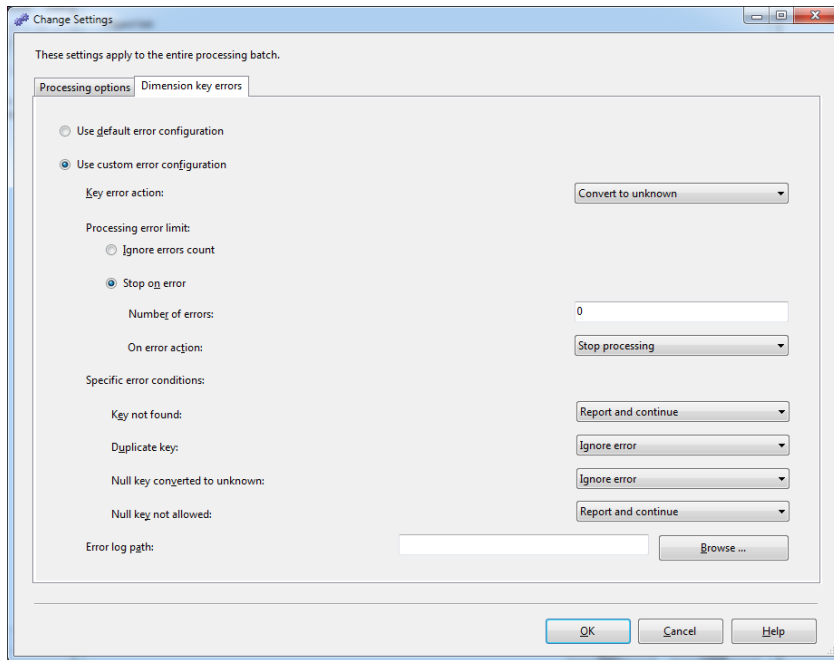


Figure 6-17. The Dimension Key Errors tab in the Change Settings dialog box of the Process Cube dialog box allows you to specify custom error behavior responses when processing a cube.

Although you have probably been processing test cubes for awhile prior to reading this chapter, you've probably gained a bit more insight into what actually happens when you run the process action. Figure 6–18 shows the output from the Process Cube dialog box. You can also direct this process information to a log file. In production, you would normally automate the cube/dimension processing via SSIS packages, using the cube- or dimension-processing tasks.

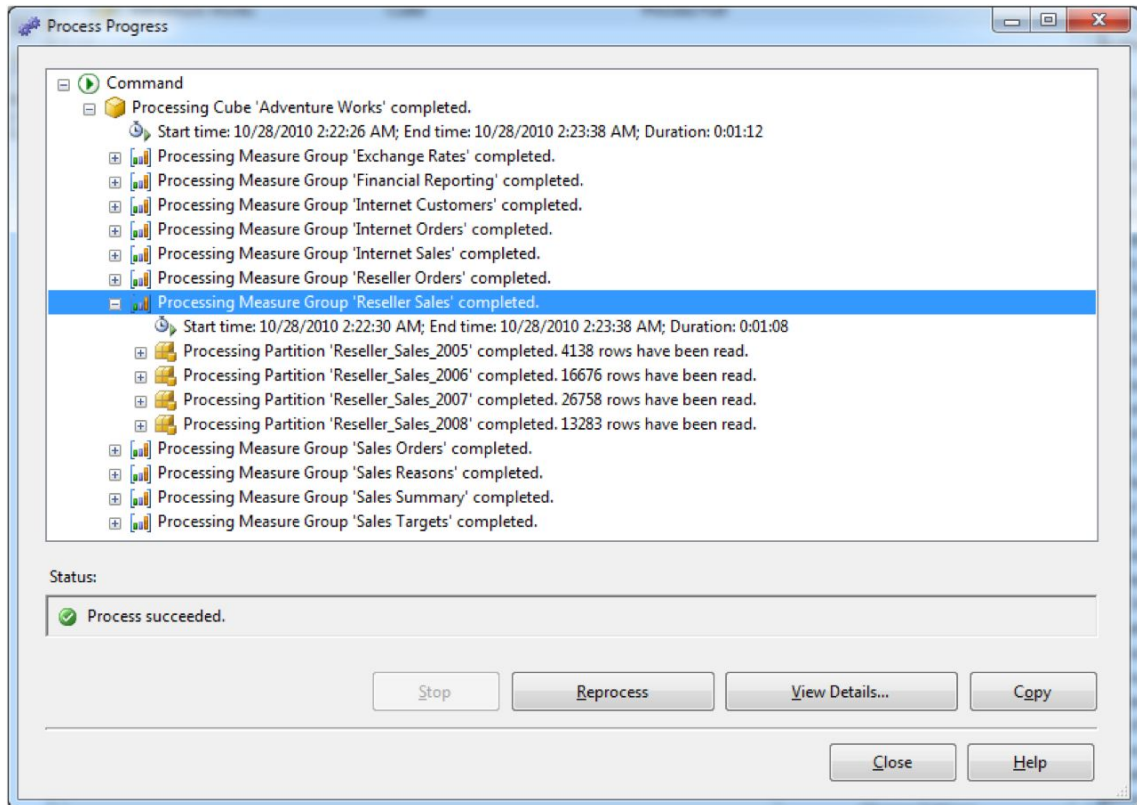


Figure 6–18. The Process Progress dialog box allows you to review the progress and duration of each step involved in processing a cube or a dimension.

Summary

This chapter covered the processes for designing physical storage for your cube. You learned about the capabilities of the three core storage modes—MOLAP, HOLAP, and ROLAP—and about aggregation design and partitioning. Remember that several of these features require the Enterprise Edition of SSAS, as noted for each feature in the text of this chapter.

The most important consideration for physical cube storage design is to get the correct balance between performance and manageability. Many customers are satisfied with MOLAP for their entire cubes, with just a bit of added custom-designed aggregations.

In the next chapter, we'll venture into the world of SSIS. You'll learn how to design basic SSIS packages, and in the process, you'll learn to use the BIDS interface for SSIS. The discussion will include the use of control flow tasks and data transformations.



Introducing SSIS

Now that you've captured the business requirements and are working on creating a basic OLAP design, the next task in creating your BI solution is to map the data from the source system(s) to your OLAP model. In larger BI projects, it is common for different people (or teams) to be working on an OLAP design and an ETL design concurrently.

Data maps are used as a basis for beginning your ETL design. As mentioned in Chapter 1, ETL stands for “extract, transform, and load.” It is important that you don't underestimate the amount of work involved in this portion of your project. A large part of the project work in a BI solution commonly consists of ETL process design, execution, and debugging. The SQL Server ETL toolset is called SQL Server Integration Services (SSIS).

Understanding ETL

ETL, in a BI project context, is the process of preparing all source data for use in an OLAP cube. The rich SSIS toolset can also be used for other types of data movement, for example, data consolidations and migrations. In this book we will focus on the BI project use of SSIS, which is extracting, transforming, and loading data from one or more source systems into an OLAP model. This process may also involve the use of an intermediate staging server that contains a staging database with staging tables. The use of a staging server and database is usually driven by the complexity and messiness of the source data. For example, in a recent project we worked on, there were 16 data sources, each with its own unique validation issues. The ETL project contained intermediate staging tables for each data source. Another consideration is whether or not the source data is contained in a relational structure. If a project will be using a large number of nonrelational source files, .csv, .txt, .xml, and so on, a staging server and database are often used as part of the ETL process.

Creating a Plan

The first part of the BI ETL process is to create a documented plan for your project's ETL. You will often create a high-level overview diagram like the one shown in Figure 7-1, which shows the data sources you will include in your project. When creating your data diagram, you should first make a list of all possible data sources for your project. A typical mistake is to not include all data sources in this selection. Your source data will most probably include one or more relational databases; however, you should not restrict yourself to relational data only. It is very common to use data from a variety of sources in the organization. These sources can include relational databases, flat-file databases, Excel spreadsheets, XML files, text files, Active Directory, ERP systems, and output from a mainframe.

You may also add detail to this diagram. Detail can include load window times (between which times you may extract data) and type of access, for example, FTP for files.

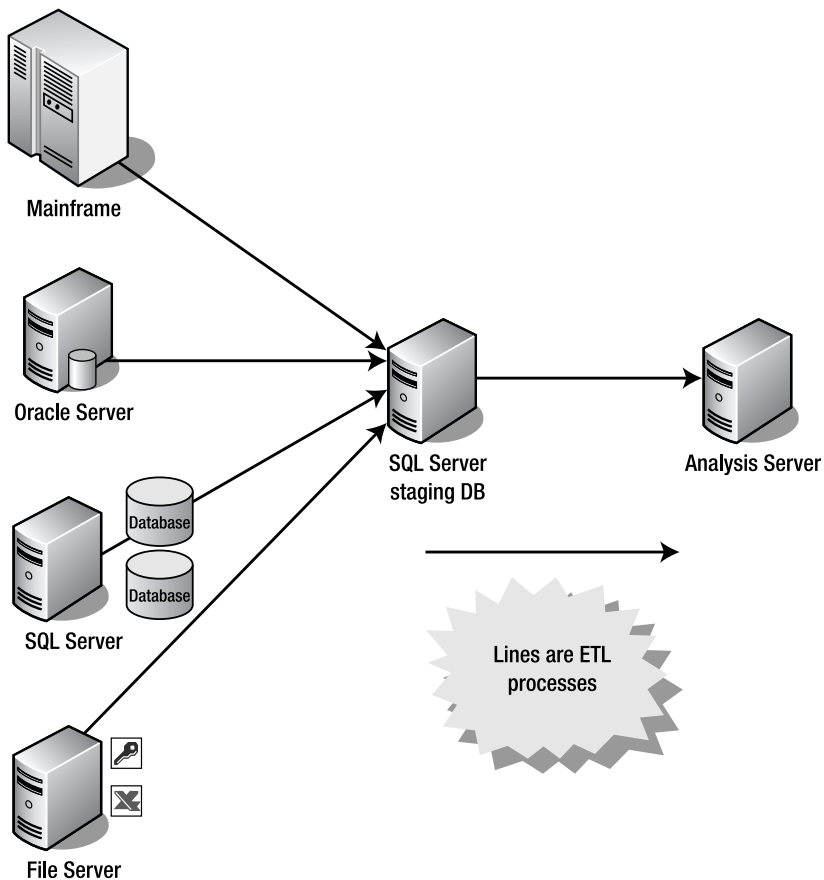


Figure 7-1. An example of an ETL data diagram

Creating a Data Map

After you've created a complete list of your data sources for extract by creating a high-level diagram and a complementary and more detailed list containing extraction information such as load windows, authentication type, transfer protocol, and so on in something like Excel, then the next step is to design a detailed data map. To do this, you will begin to map the data sources more specifically to the data destination locations. The destinations are based on the OLAP star schemas that you've already created and may include map-through staging tables, if you've chosen to use them. As you continue your design documentation, you will add more detail to the information captured in this diagram. The outcome of this process is called a *data map*.

The use of a staging server, although not a requirement for the use of Analysis Services, is a common practice. In addition to choosing to add this dedicated server due to data load complexity, another consideration is that you can offload work from the production OLTP server and consolidate ETL processes onto one physical machine. You may not have a one-to-one match for some of the tables or the table fields at this point. At this early stage, you are beginning to match source data with destination data. Later, you will identify SSIS transformations and map from source to destination all the way to the column level for each table of each data source. An example of a portion of an early data map is shown in Figure 7-2. Within the three sections in this figure—sources, transformations, and destinations—boxes represent items that will need to be mapped; that is, there is an .xls source for customers, and that .xls source will need a concatenation transformation and will be sent to the DimCustomer table destination.

You will commonly work with subject matter experts (SMEs) to make corrections in your OLAP star schema based on new information discovered during the ETL mapping process. You might also need to work with SMEs concerning nonintuitive source information, such as obscure column names in source tables or data that must be translated from a numeric to a text value to be meaningful. Sometimes, this will result in your OLAP star schema design changing, and sometimes, this will also result in the inclusion of more source data than was originally planned for. All stakeholders should approve any significant changes to the scope of your BI solution. These changes may also cause budget and resources to be adjusted for your BI project.

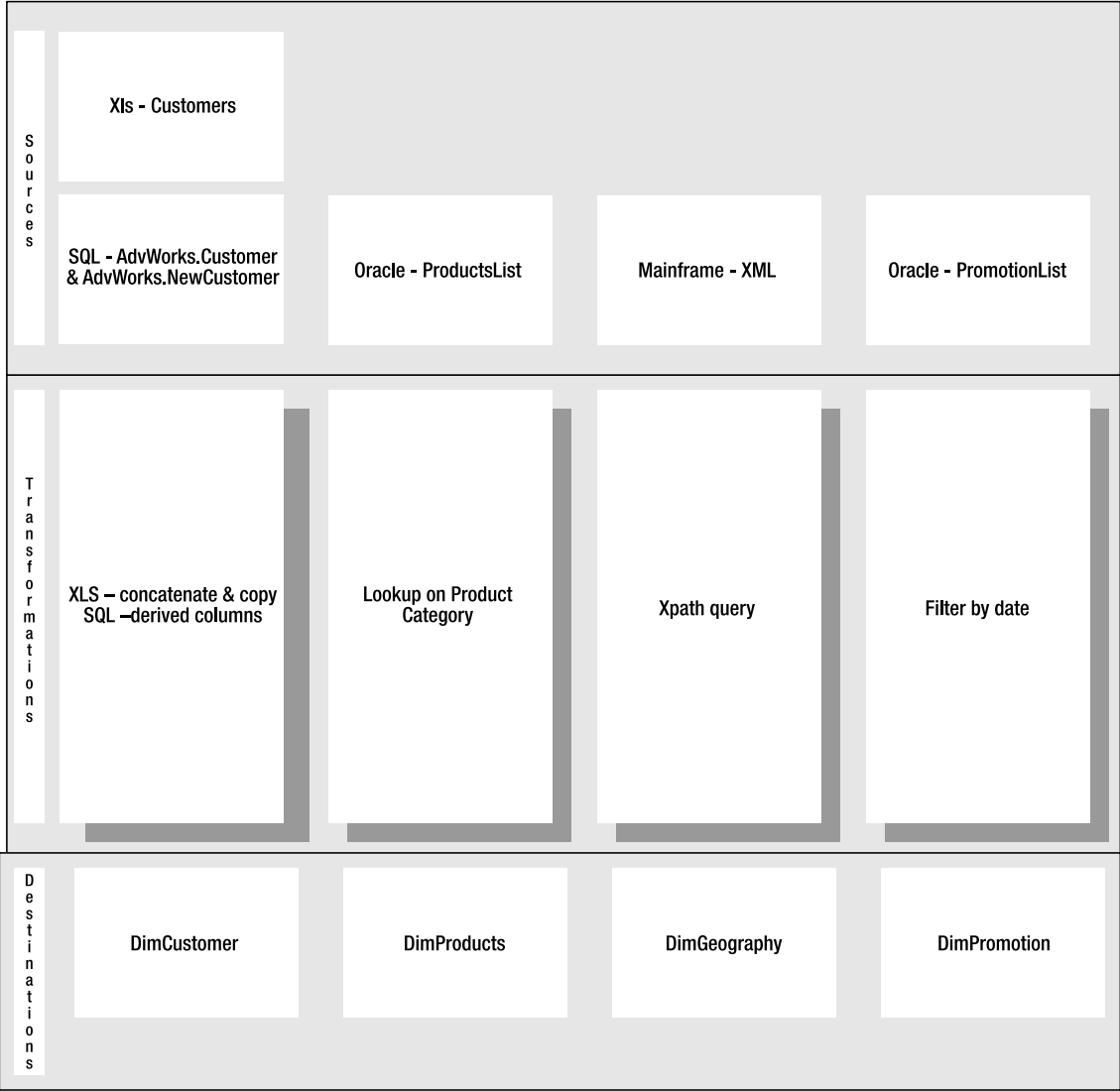


Figure 7-2. You should create a high-level mapping of data sources to destinations, including partial transformation information during the early stages of creating your BI project data map.

Refining a Data Map

The next step in your ETL-mapping process is to create a more detailed data map. We’ve used Excel for this in past BI projects. Your goal here is to add as much detail as possible to your data map prior to starting development of SSIS packages. Figure 7-3 is an example of a more detailed data map,

sometimes referred to as *Source to Target*. Note the addition of all columns in the destination table and a partial mapping of sources to destinations at the column level. Strive to be as complete as possible in your data map prior to starting actual SSIS package development.

| Dimension Table Name | Dimension Column Name | Constraint Type | Surrogate Key | Exist in OLAP Only | Data Type | Original Data Source FC |
|----------------------|-----------------------|-----------------|---------------|--------------------|-----------|---|
| StaffDim | NewStaffID | PK | Yes | Yes | | Academy.Member.MemberID |
| | OldID | | | | | Academy.Member.LastName |
| | LName | | | | | Academy.Member.FirstName |
| | FName | | | | | Academy.Member.Email |
| | Alias | | | | | Academy.Role.Title |
| | Role | | | | | if Academy.Role.Title = visitor then true |
| | IsVisitor | | | | bit | Unknown |
| | SalesLocation | | | | | Unknown - EDS |
| | SubRegion | | | | | Unknown - EDS |
| | Region | | | | | Unknown |
| | Area | | | | | Unknown |
| | BigArea | | | | | Unknown |
| | ClothingCut | | | | | Academy.StaffProfile.ProfileStatusXml.ClothingCut |
| | ShirtSize | | | | | Academy.StaffProfile.ProfileStatusXml.ShirtSize |
| | PictureUrl | | | | | Academy.StaffProfile.ProfileStatusXml.PictureUrl |
| | BioText | | | | | Academy.StaffProfile.ProfileStatusXml.BioText |
| | BirthDate | | | | | Academy.StaffProfile.ProfileStatusXml.BirthDay+BirthMonth |

Figure 7–3. More detailed data mapping, showing a single destination table, with source data listed for each column

Adding a Staging Server

In Figure 7–1, the physical implementation of a BI solution shows a separate physical server as a staging server. Although this separate physical server is not required for all BI solutions, you should consider the factors that might cause you to add this server to your solution. Although you can easily install both SQL Server, which includes SSIS and SSAS, on the same physical server, there are many good reasons for separating these two products. In most production BI solutions that we've implemented, separating SSIS and SSAS has been selected as the final production configuration. By using two servers for SSIS and SSAS, you are separating two key components of your BI solutions: the ETL processes and the cube processing/hosting environments. In doing this, you are creating a configuration that is easier to maintain, performance tune, and scale than using one server for both types of functionality.

Although the two-server configuration is the one we've most commonly used in production, it is perfectly valid to install both SSIS and SSAS on the same physical server for development or testing. Of course, it is always preferred to duplicate the production environment in the testing and development areas, but the reality for many of my customers is that their resource constraints necessitate simpler solutions in testing and development.

After you decide the appropriate physical configuration given your particular requirements and constraints, and have a relatively complete data map, you are ready to begin the actual ETL process development for your BI project. You will usually use SQL Server SSIS to do this. If your source data is exceptionally clean, it is theoretically possible to use a simpler method, such as T-SQL statements (if all source data was contained in SQL Server), rather than SSIS packages to perform the ETL portion of your BI project. This is theoretical because although a couple of customers have suggested this simplified approach, we've not found a business environment yet that was sufficiently disciplined to provide data that was clean enough to be directly used in a BI project.

Creating a Basic SSIS Package

SSIS packages contain four fundamental parts: control flow, data flow, error handlers, and configurations. Although you can design packages with these four fundamental parts in BIDS, we're not going to start there just yet; rather, you'll use the included Import and Export Wizard to design your first package. There are three reasons for doing this: simple data movement is quicker; wizard-designed packages can serve as templates or starting points for other packages, and the wizard encapsulates quite a bit of functionality.

■ **Note** This section has you accessing the Import and Export Wizard from within SSMS. You can also access the Import/Export Wizard from inside of BIDS by opting to create an Integration Services Project, right-clicking the SSIS packages folder in the Solution Explorer window, and choosing SSIS Import and Export Wizard.

To access the wizard, right-click your AdventureWorks2008R2 database in the Object Explorer window in SSMS, and click **Tasks ► Import Data**, or click **Tasks ► Export Data**. This will start the wizard. After you click **Next** on the Welcome dialog, you will see the Choose a Data Source dialog, shown in Figure 7-4.

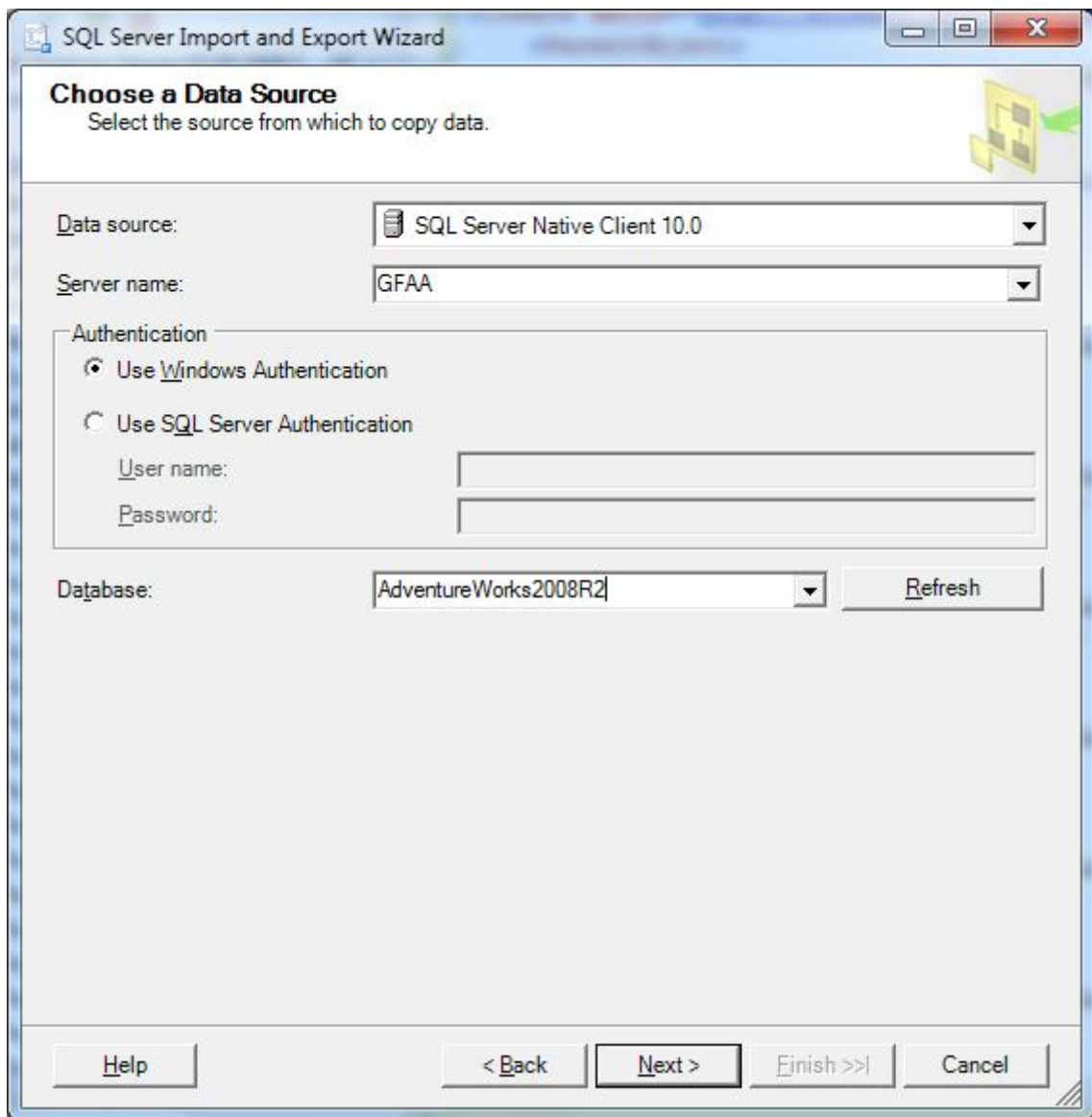


Figure 7–4. Choosing a data source is the first step in using the SQL Server Import and Export Wizard for SSIS.

The next step is choosing a destination for your data. For this basic package, choose `tempdb` as your destination database. The wizard can work with SQL Server and non-SQL sources and destinations. After you've selected both source and destination, you'll need to choose one of two options: "Copy data from

one or more tables or views” or “Write a query to specify the data to transfer” (which copies some subset of data to your selected destination), as shown in Figure 7–5.

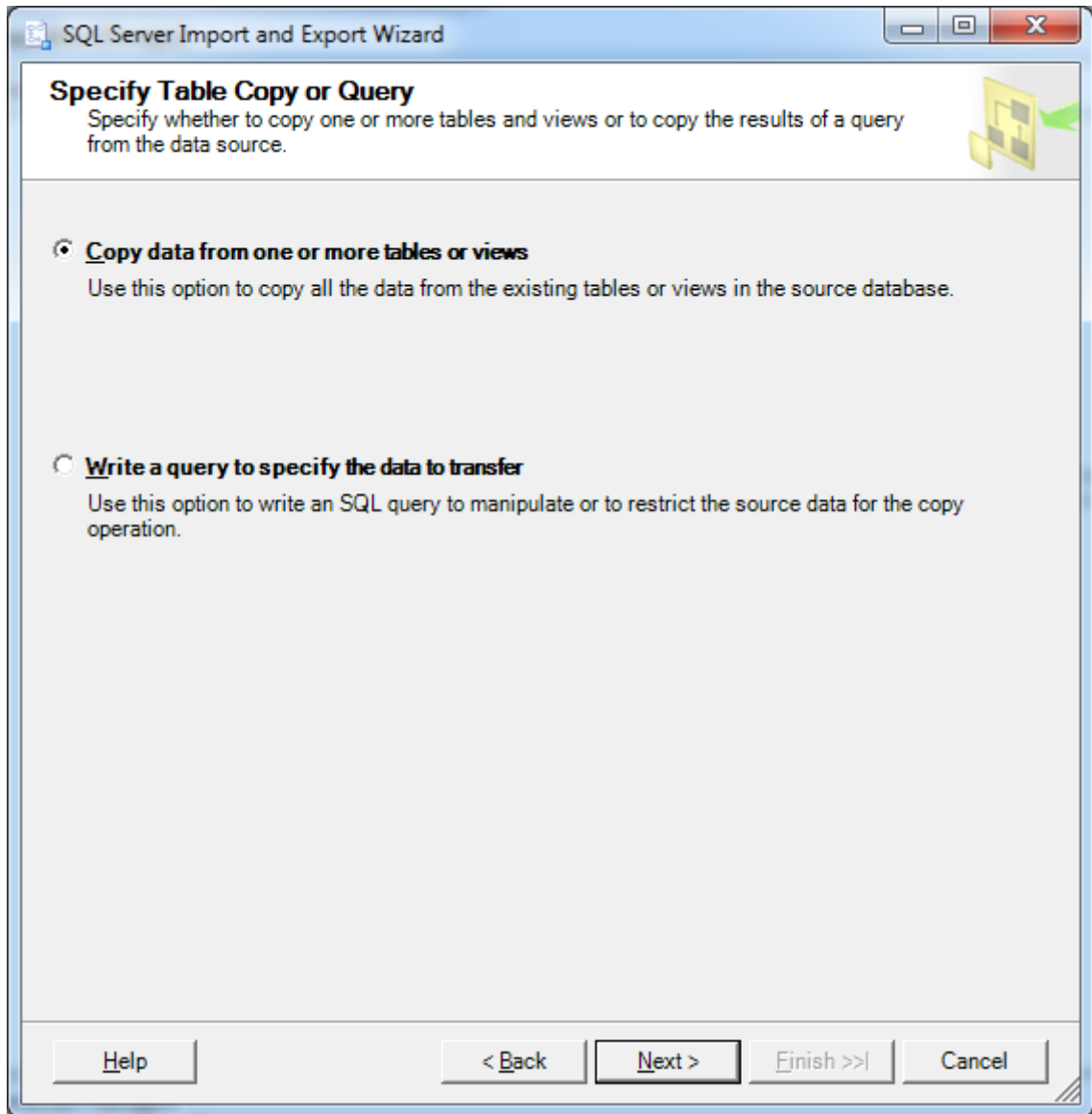


Figure 7–5. In this step of the Import and Export Wizard, you select from one of the two shown options.

Select the first option, and click Next. In the Select Source Tables and Views dialog, choose the [Person].[AddressType] table. The wizard will allow selecting one or more tables or views from the

source location. The default is to copy the object metadata (or structure) and data to the destination location in the same format as the source. You can adjust this, however, by clicking the Edit Mappings button. If you do this, you can skip importing selected columns by setting the Destination column value to <ignore>. This is shown for the column named rowguid in Figure 7-6.

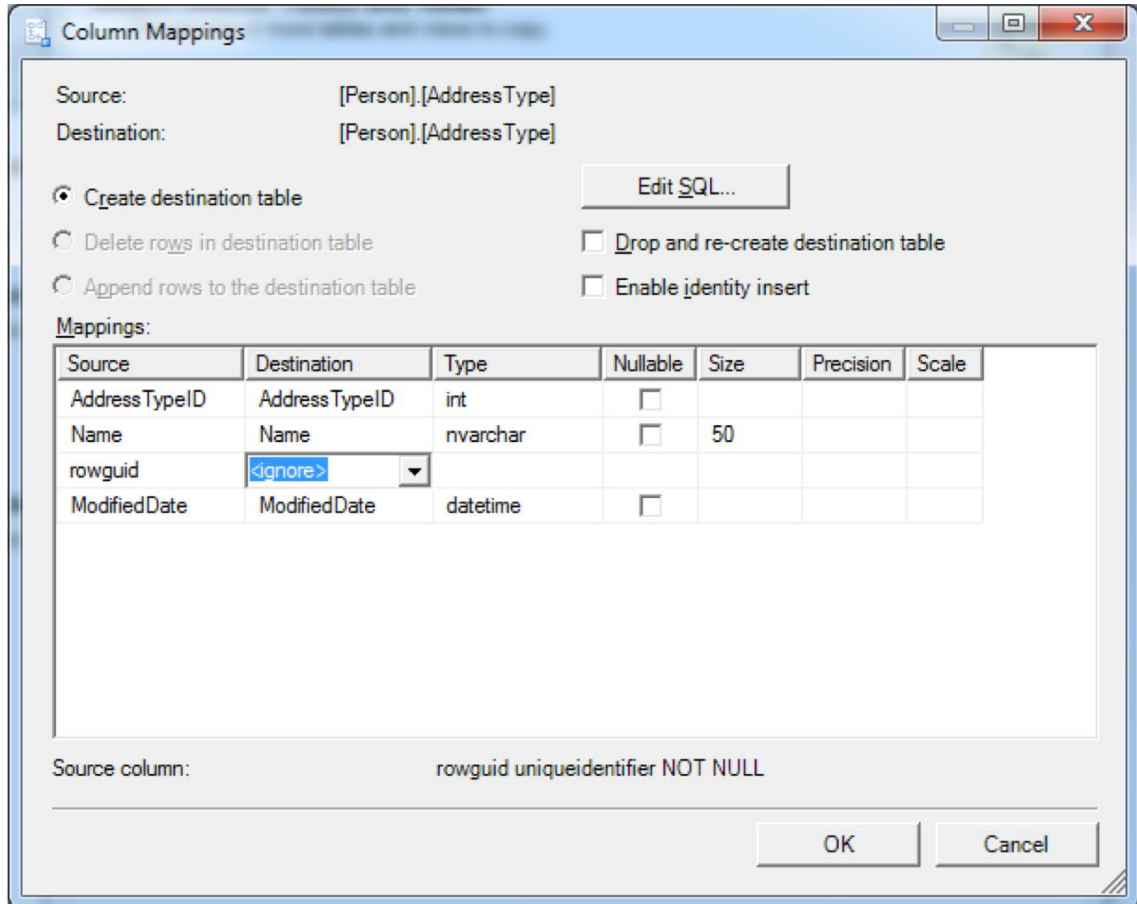


Figure 7-6. In this step of the wizard, you can set the package to skip or ignore selected columns.

After you click OK and Next, the wizard allows you to run the import (or export) process that you've configured immediately, and (optionally) to save it as a SSIS package. You can select from two locations if you want to save your package: SQL server or the file system.

If you store your package on SQL server, you will have to specify a server name and authentication credentials. The package will be stored in the MSDB database on the server you selected. If you select storage on the file system, you must supply a path. The package will be stored at the selected location as a file with a .dtsx extension. Although not required, the preferred method is to store your packages on a SQL Server instance in MSDB, so that you can more easily attend to administrative tasks, such as performing backups, and scheduling execution. At this point, you want to uncheck "Run immediately", check Save SSIS Package, and click Next to load the Save SSIS Package dialog, as shown in Figure 7-7.

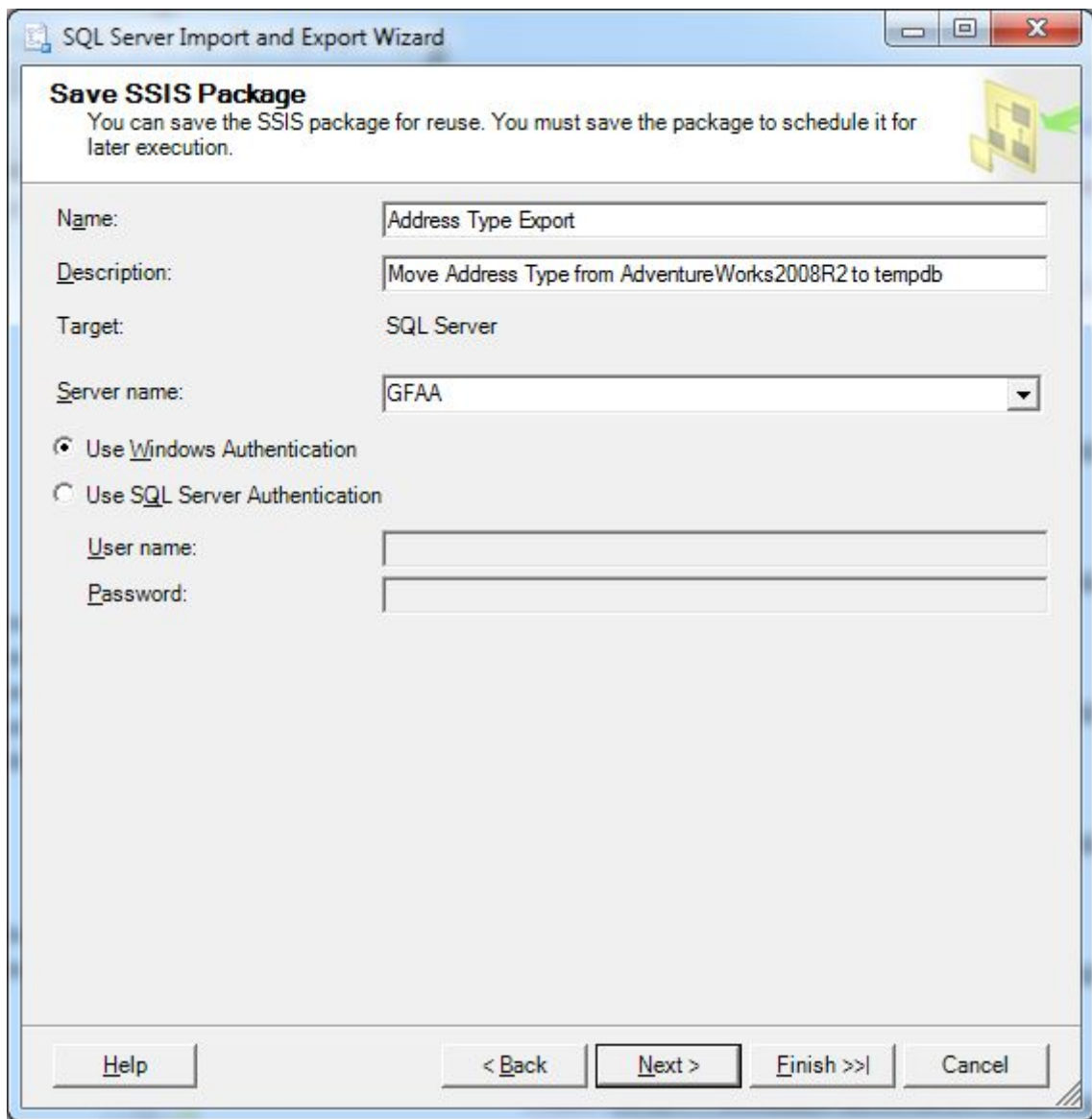


Figure 7-7. In the final step of the wizard, you can save and store the package on SQL Server.

Give your package a name and description, and click Next to save your package. Finally, click Finish to complete the wizard.

If you want to run your package, you can run it directly from SSMS. To run your package from SSMS, click Connect ► Integration Services in Object Explorer. After you've successfully connected, you'll see the package in the menu tree by navigating to Stored Packages ► MSDB. Right-click your package

name, and choose Run Package. In the Execute Package Utility dialog, you can configure several runtime properties, including error log locations and connection information for the package, as shown in Figure 7–8.

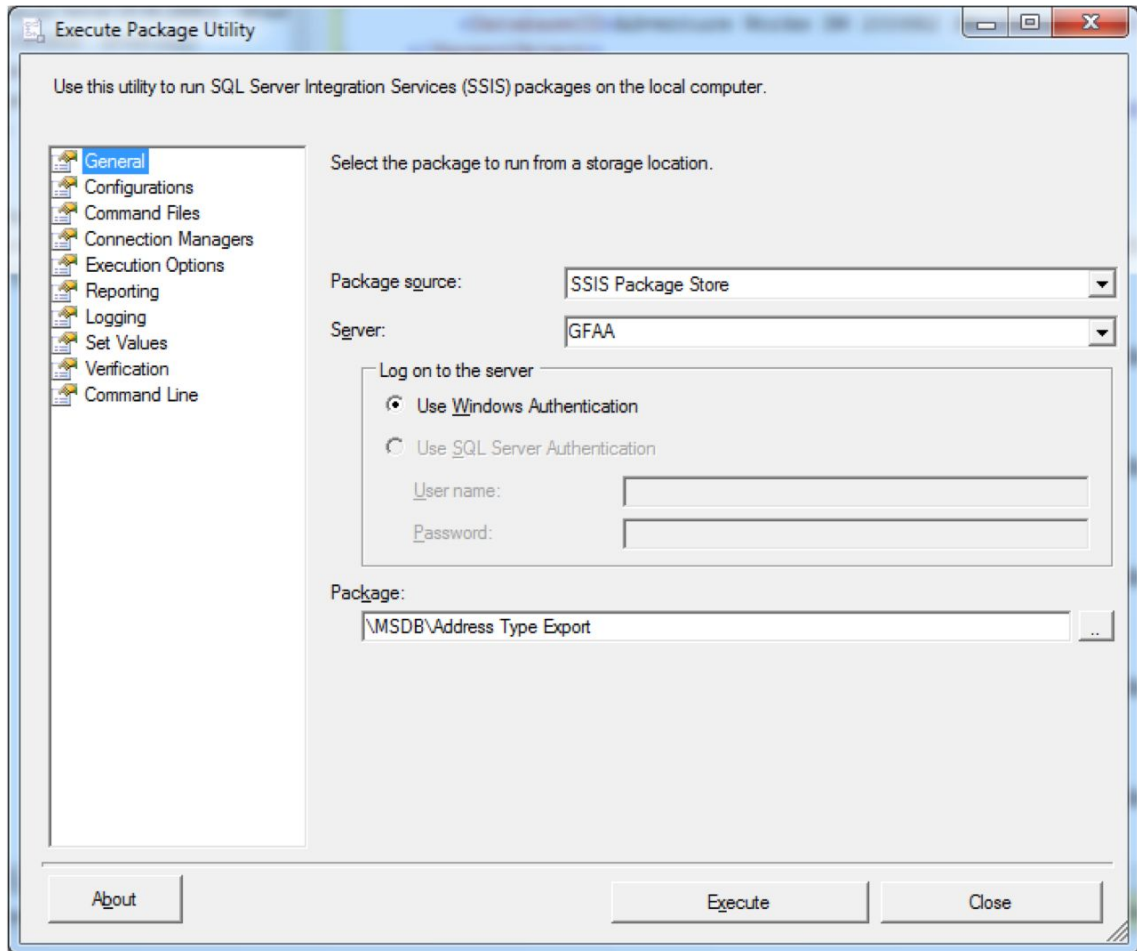


Figure 7–8. In SSMS, you can set many runtime properties of a package.

One reason to use the Import and Export Wizard in SSMS rather than using BIDS to create simple packages is to quickly create prototype packages (templates as well) that you can later edit using BIDS.

Building Basic SSIS Packages

As with other types of BI design, such as cubes, you'll open BIDS to begin creating an SSIS package. BIDS functionality is accessed via a set of included BI template types that are available inside of the Visual Studio (VS) interface. BI objects are created using the BIDS templates, including templates to create

SSAS databases (cubes, dimensions, and data mining structures and models), SSIS projects (SSIS packages), and SSRS reports and report models. Templates consist of starter folders and files that are specific to the project type selected. To start working in BIDS, you open it from the Start menu, click **File ► New ► Project**, and then select the appropriate project type from the templates of type **Business Intelligence**. Solutions are containers for projects, and projects are containers for files and folders.

Developing SSIS Packages

As with designing your first cube, developing SSIS packages requires that you use BIDS to create a new project. In this case, you'll select a project from the **BI** category and of type **Integration Services Project**. After you do that, you'll see the development environment shown in Figure 7–9.

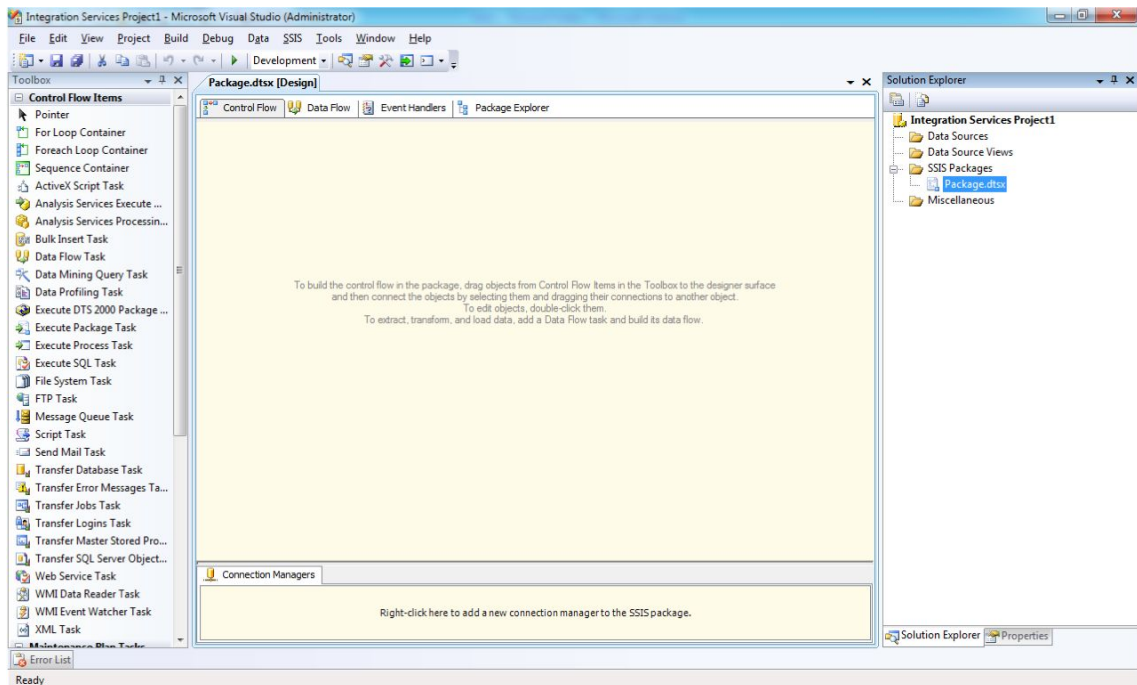


Figure 7–9. SSIS packages are authored using BIDS with a project type of *Integration Services Project*.

The first section you'll use in BIDS is the **Solution Explorer**. This shows the name of the project and is, in essence, a container for multiple **.dtsx** packages. Projects are a natural way to group packages together for doing the ETL processes for a particular BI project (or SSAS database, which will typically contain one or more OLAP cubes).

The nodes, or folders, below the project name are provided in Table 7–1.

Table 7–1. List of Folders in BIDS for SSIS Package Template

| Folder | Purpose |
|-------------------|--|
| Data Sources | Contains connections used in one or more packages (global connections) |
| Data Source Views | Contains abstractions against one or more data sources |
| SSIS Packages | Contains SSIS packages (.dtsx files) |
| Miscellaneous | Contains other file types, that is, .txt, .xml, and so on |

If you wanted to edit a package that you had created in SQL SSMS using the Import and Export Wizard, you could import it into BIDS by right-clicking the SSIS Packages folder in Solution Explorer and choosing Add Existing Package. In the Package location drop-down list in the Add Copy of Existing Package dialog, you choose SSIS Package Store (see Figure 7–10).

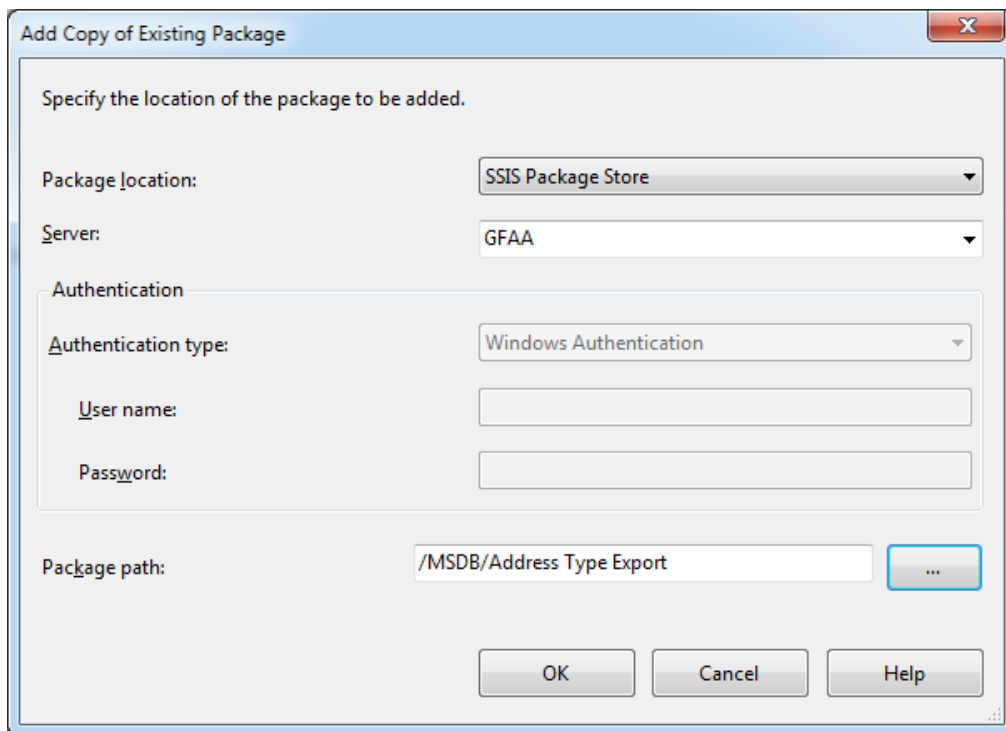


Figure 7–10. In the Add Copy of Existing Package dialog, you select the import package from whatever SSIS Package Store location you’ve originally stored your package(s) in to be able to edit packages created in SSMS using the Import and Export Wizard.

Designing SSIS Packages

Now, you are ready to design your first SSIS package. Using your detailed data map, you should have information about the source, destination, and potential transformations documented. This will be the basis for your package design. For BI projects, best practice is to design one or more packages per destination table and not design individual packages that load data into multiple tables. By following this simple guideline, and ideally documenting it as a standard for your ETL team, you'll be reducing the effort involved in updating and maintaining your SSIS packages over the lifetime of the BI solution.

For our first example, you'll create a simple package that extracts data from a single source location and loads it into a staging (or intermediate) table in your star schema database. The first thing to consider is how you'll connect to the data sources.

Configuring Connections

Although it is possible to define connections only within each package, you will most often reuse connections across multiple packages within the same project. To create a new connection, right-click the **Data Sources** folder in the Solution Explorer, and choose **New Data Source** to open the Data Source Wizard. In this wizard, you can either reuse a previously defined connection for any other SSIS package designed on your computer, or you can create a new connection. Click **Next** at the Welcome dialog and then click **New** in the "Select how to define the connection" dialog to open the Connection Manager, shown in Figure 7-11.

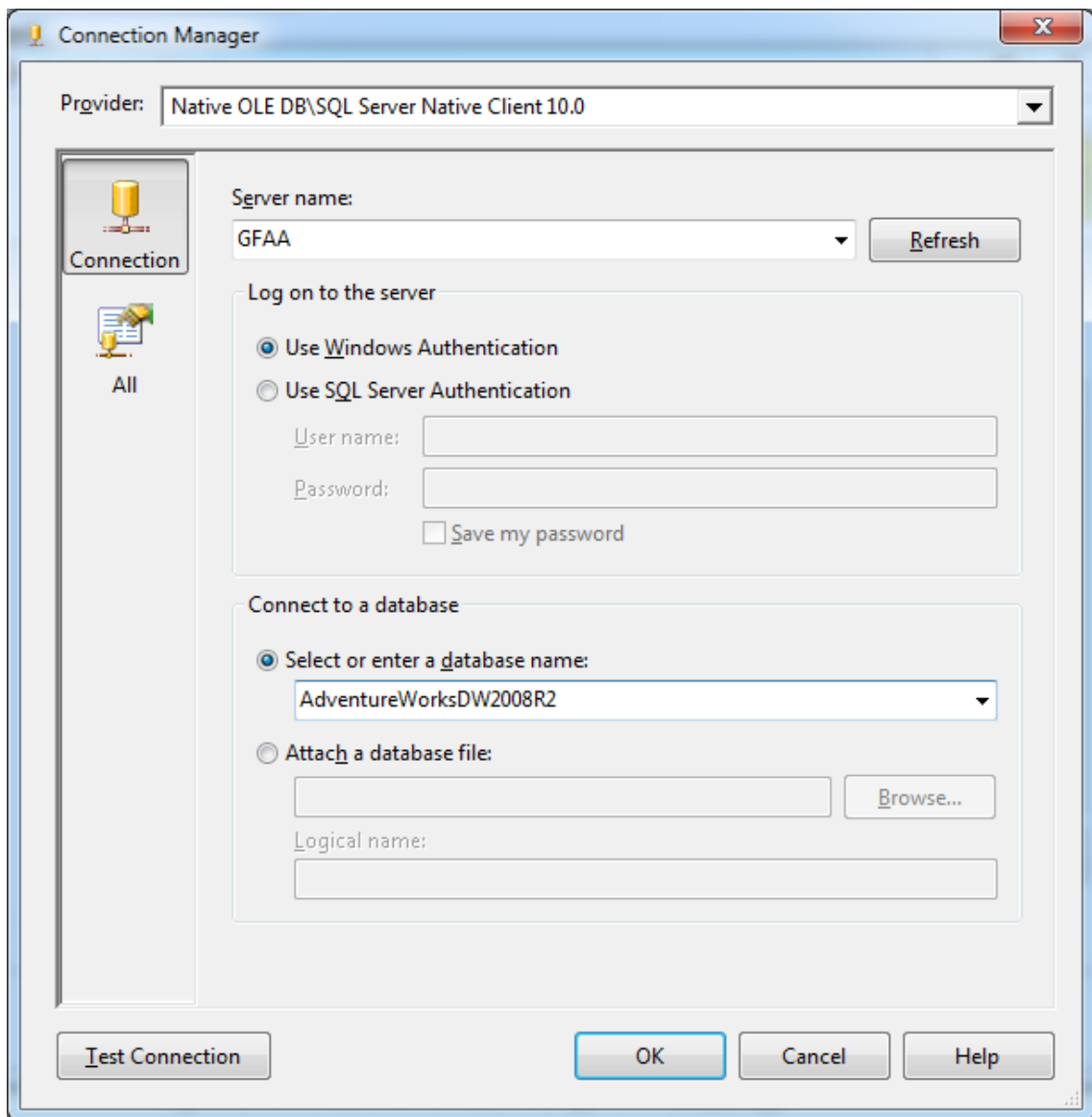


Figure 7–11. In the Connection Manager dialog in BIDS, you configure reusable connections. The All button on the left allows you to set advanced connection properties, such as connection timeout values.

The next step is to associate this global connection with the specific package that you are working on. To do this, right-click the bottom-center area of the package design surface in BIDS titled Connection Managers; a menu of connection types appears, as shown in Figure 7–12. Click New

Connection from Data Source, and select the connection name that you previously created. You will now be able to associate this connection with tasks in your particular package.

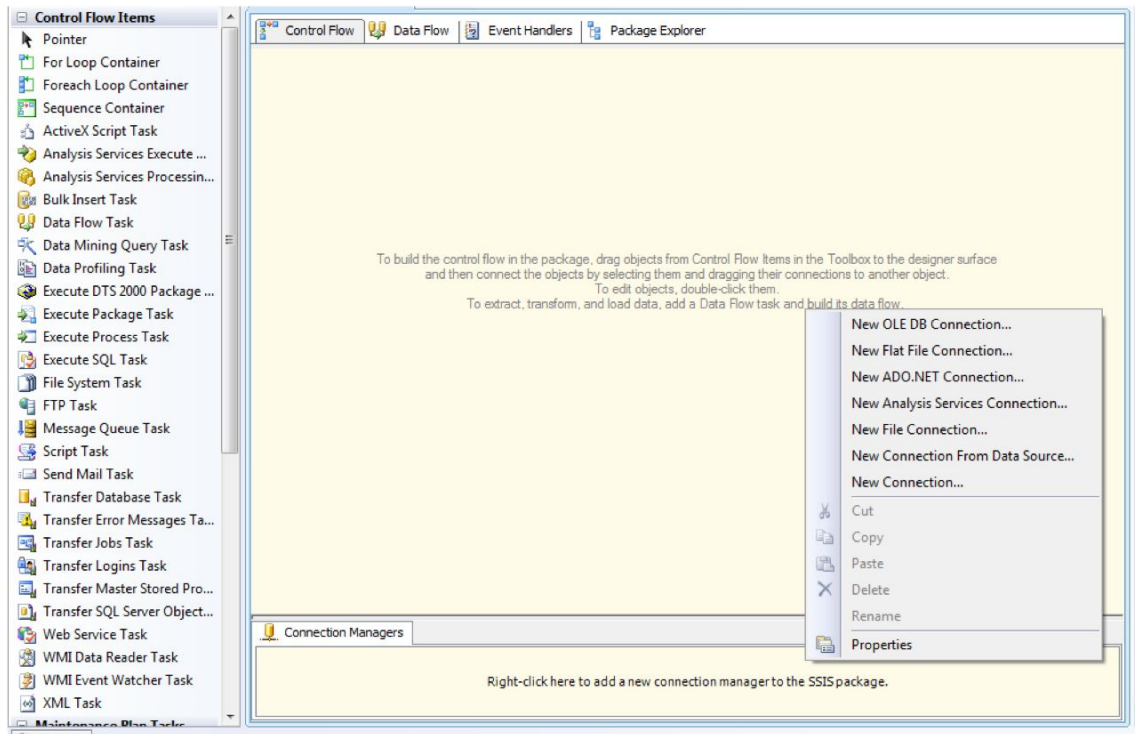


Figure 7–12. Menu of data source types to select from.

To add a connection specific to the particular package that you are designing and not globally available to all packages in your current project, right-click in the Connection Managers area, click the context menu item that reflects the provider (that is, OLE DB, ADO.NET, and so on), and then configure the connection as per your requirements. Figure 7–12 shows the complete list of available providers in a default installation of SSIS. You can see and work with this list by selecting New Connection from the context-sensitive menu that is available after you right-click in the Connection Managers design area for SSIS packages in BIDS.

Adding Control Flow Tasks

Your next step in package design is to select one or more control flow items from the toolbox and drag them to the Control Flow design surface section. The Control Flow surface is a workflow manager for your SSIS packages. The control flow design surface allows you to compartmentalize a package by separating different types of tasks and data flows. Individual data flows can separate data movement and transformations by unit of work or line of business. Figure 7–13 shows the toolbox for the control flow items. You can see by looking at the names of the items in Figure 7–13 that *control flow* refers to items that perform some sort of action (most all of the task names are self-explanatory). Books Online (BOL)

offers explanations of the functionality associated with each item. We will review some of the more complex control flow tasks in Chapter 8, “Intermediate SSIS,” and Chapter 9, “Advanced SSIS.”

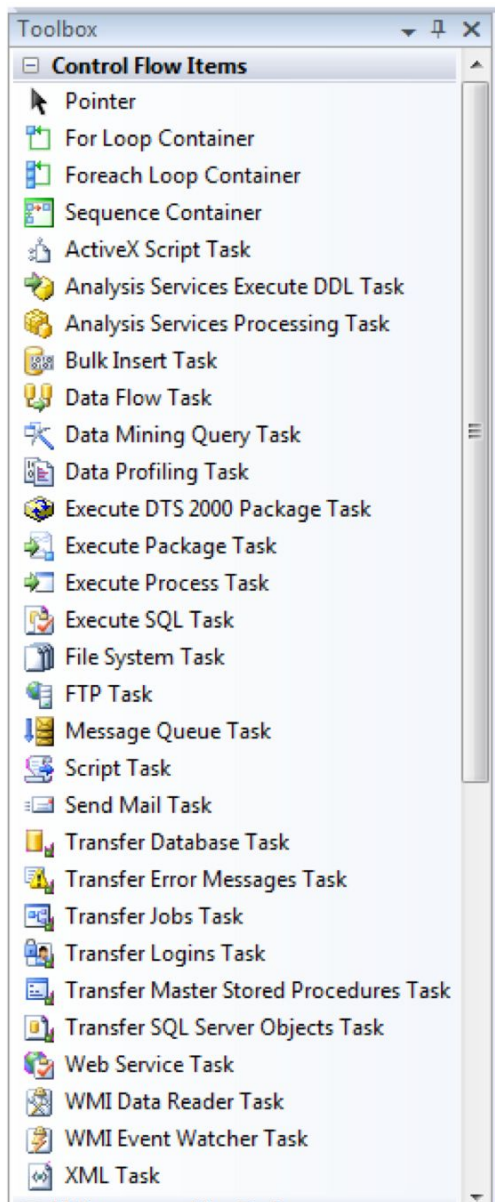


Figure 7–13. The Control Flow Items section in the BIDS toolbox displays a variety of stock tasks to use in your SSIS packages.

For your first package, you will use just two of the tasks from this toolbox: Data Flow Task and Send Mail Task. Drag each of the items to the center work area, or designer surface, in BIDS, placing the Data Flow Task just above the Send Mail Task. Click the Data Flow Task, and drag the green line from the Data Flow Task to the Send Mail Task until the two tasks are connected. You'll notice that the Send Mail Task has a red warning icon on it. If you pass your mouse over the error icon, a tooltip will tell you that the "SMTP server is not specified," so the task cannot execute. To review the cause of this error, right-click the task and select **Edit**. Next, select **Mail** from the left-hand pane and you'll be presented with the Send Mail Task Editor configuration dialog, an example of which is shown in Figure 7-14.

So far, you've created a package that will complete some kind of data movement or flow and will send an email via SMTP on successful completion of that data flow task (after you add the SMTP server location information). For the purposes of this quick discussion, we aren't going to complete that configuration. The next step in the design of this pseudo-package is to configure the data flow itself.

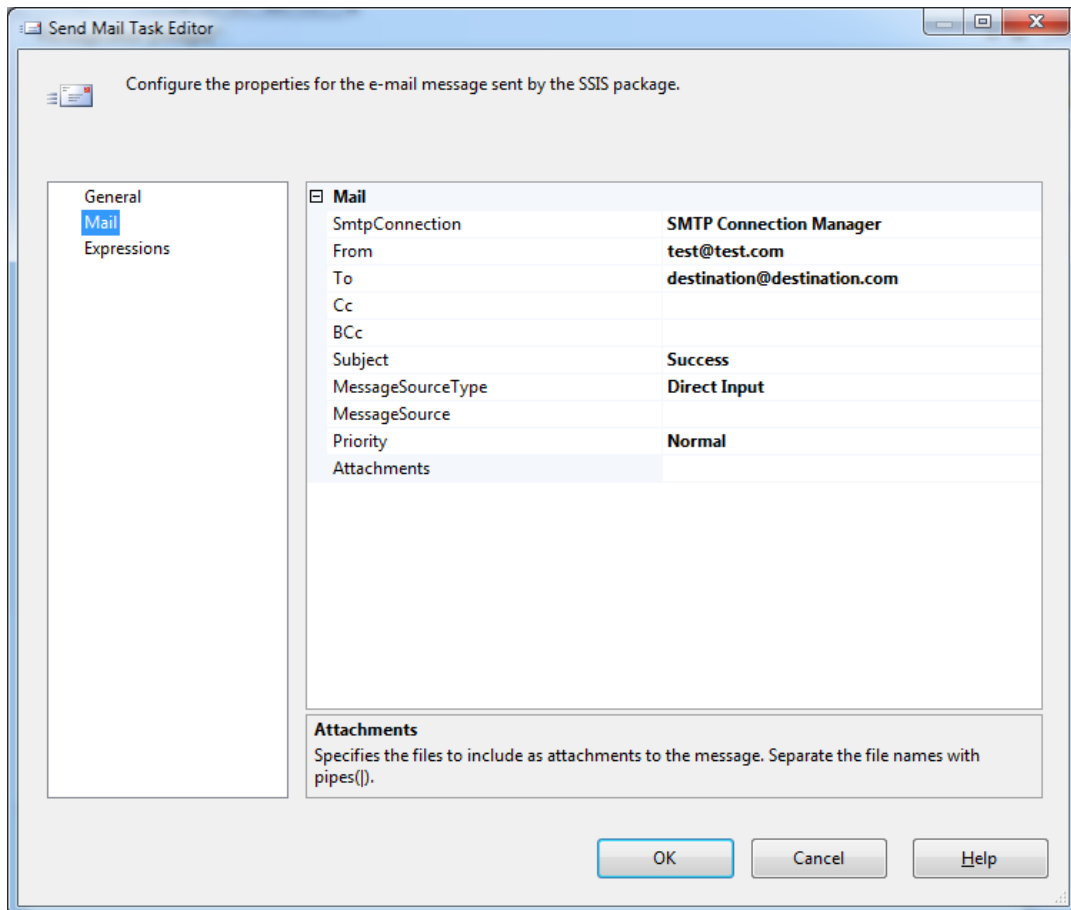


Figure 7-14. The Send Mail Task Editor dialog box requires you to configure a SMTP connection and to supply from and to information.

Configuring Data Flow Tasks

To begin configuring your Data Flow Task, double-click the Data Flow Task on the Control Flow design surface. This will take you to the Data Flow design surface for that particular Data Flow Task (see Figure 7-15).

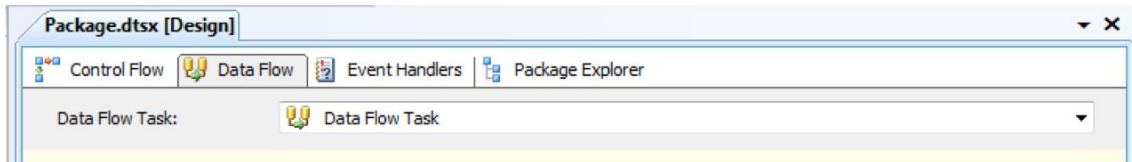


Figure 7-15. The Data Flow work area in BIDS allows you to visually configure a data flow for a particular Data Flow Task in the control flow area of your SSIS package.

You will usually select at least one data source, one transformation, and one destination from the toolbox. You do not need to add any transformations to a data flow; that is, you can simply add a data source and a data destination. Figure 7-16 shows all data sources and data destinations from the BIDS toolbox.

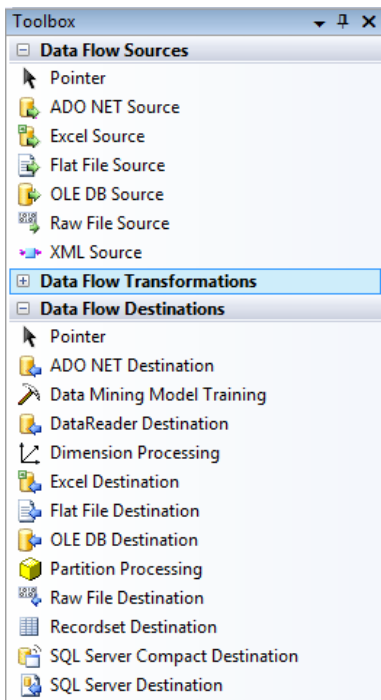


Figure 7-16. Data Flow Sources and Data Flow Destinations from the toolbox in BIDS reflect a variety of interoperability between different types of data sources and data destinations in SSIS packages.

For this particular example, you'll simply drag on an OLE DB Source and an OLE DB Destination. To configure each OLE DB item, right-click and choose Edit. Next, associate your OLE DB Source with the existing AdventureWorks connection, and choose DimEmployee as your source table. After you've completed this step, drag the green arrow from the OLE DB Source to the OLE DB Destination. Open your OLE DB Destination, and click New to create a new destination table. Click OK in the Create Table dialog to instruct SSIS to create a [OLE DB Destination] table, and set it as the data flow destination. Finally, click Mappings in the left-hand pane to have SSIS automatically map source to destination columns by name, and click OK. In this case, no transformations were added between the source and destination. Although atypical in the real world, this example demonstrates the simplest possible data flow, and is shown in Figure 7–17.

■ **Tip** When should you use an ADO NET source (or destination) and when should you use an OLE DB source (or destination)? Use an ADO NET when consuming or providing data to any .NET data provider. Use an OLE DB source (or destination) when consuming or providing data to any OLE DB provider.

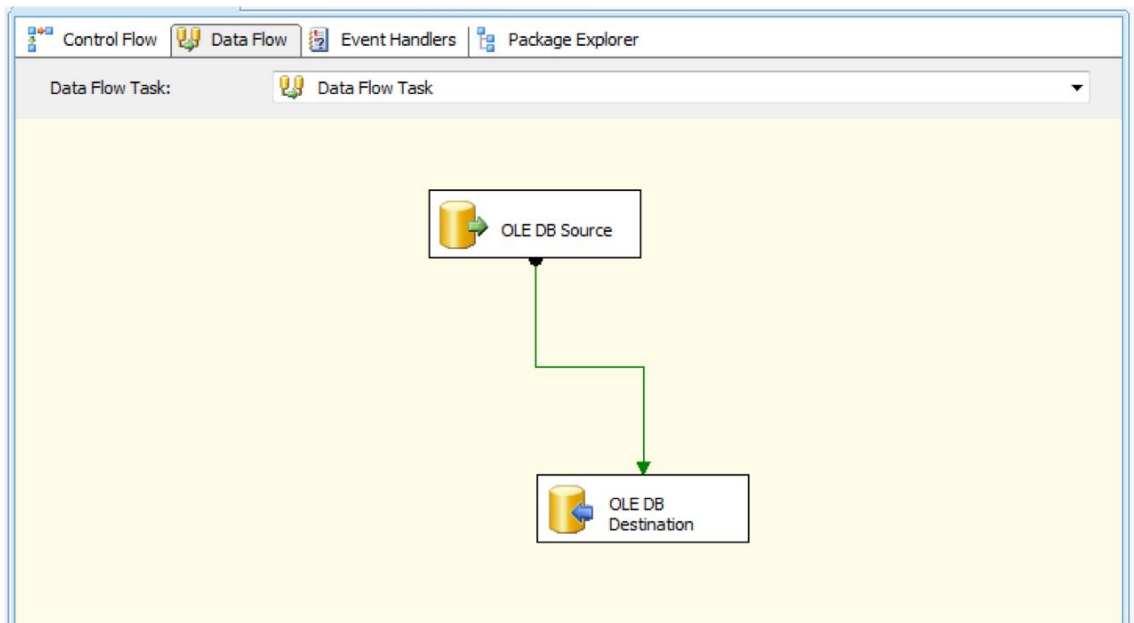


Figure 7–17. Configure data flows by using data sources and destinations in the DataFlow work area of SSIS.

To test your first package, right-click the package name in Solution Explorer, and choose Execute Package. BIDS not only allows you to execute packages but it shows execution status by changing the color of the source and destination items (as well as any transformations—this item type is not shown in this example): yellow for executing, green for success, and red for failure. Row counts are also shown on

the designer surface. In addition to this, SSIS contains a variety of debugging techniques. We'll discuss these in detail in Chapter 9. Figure 7–18 shows the result of a successful execution of this sample package.

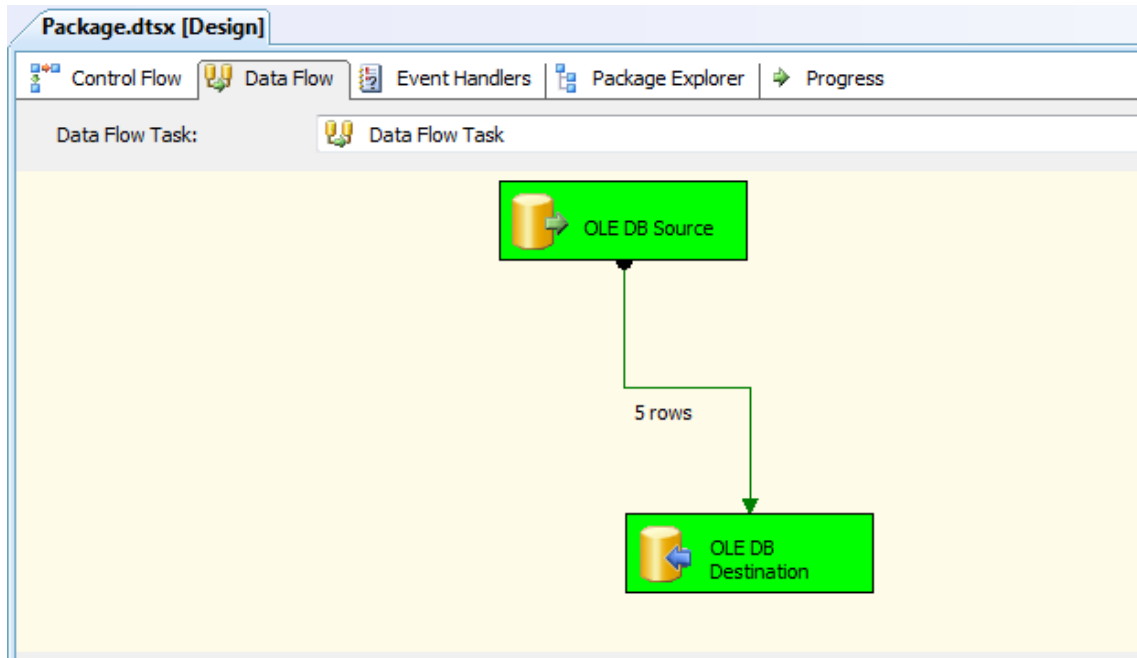


Figure 7–18. BIDS allows you to execute your SSIS packages, and the design environment indicates data flow status and row count on the design surface.

Adding Transformations to the Data Flow

Using your data map, you'll want to use one or more of the supplied data transformations in your package. Figure 7–19 shows the possible options.

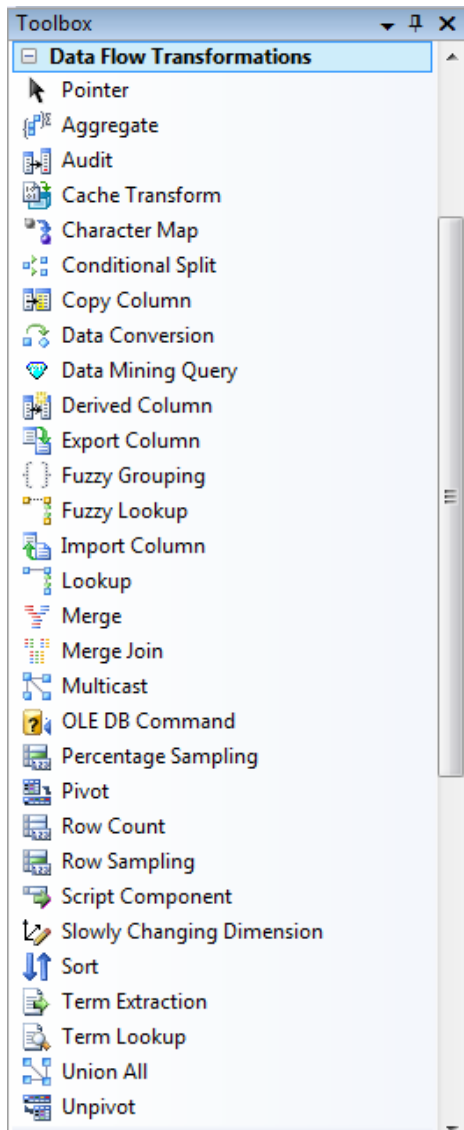


Figure 7–19. SSIS has a large array of Data Flow Transformation components to select from.

To enhance your basic package, add two transformations to the data flow: Derived Column and Sort. Continue by removing the data flow path (green line) between OLE DB Source and OLE DB Destination. Click your OLE DB Source, and drag the green data flow path (success path) to the Derived Column transformation.

Configure the Derived Column transformation by right-clicking the transformation on the design surface, and choosing Edit. The Derived Column dialog box appears in which you can create the new column using SSIS expression syntax. For more information about the capabilities of SSIS expression syntax, see the BOL topic “Syntax (SSIS).” You’ll note that all columns and variables available in the data flow are shown in the Derived Column Transformation Editor dialog box. Also, there is a function reference. To create the new column, complete the expression field. For this transformation, enter `FullName` as the Derived Column Name, `FirstName + " " + LastName` as the Expression, and click OK. These entries simply concatenated the `FirstName` and `LastName` columns, as shown in Figure 7–20.

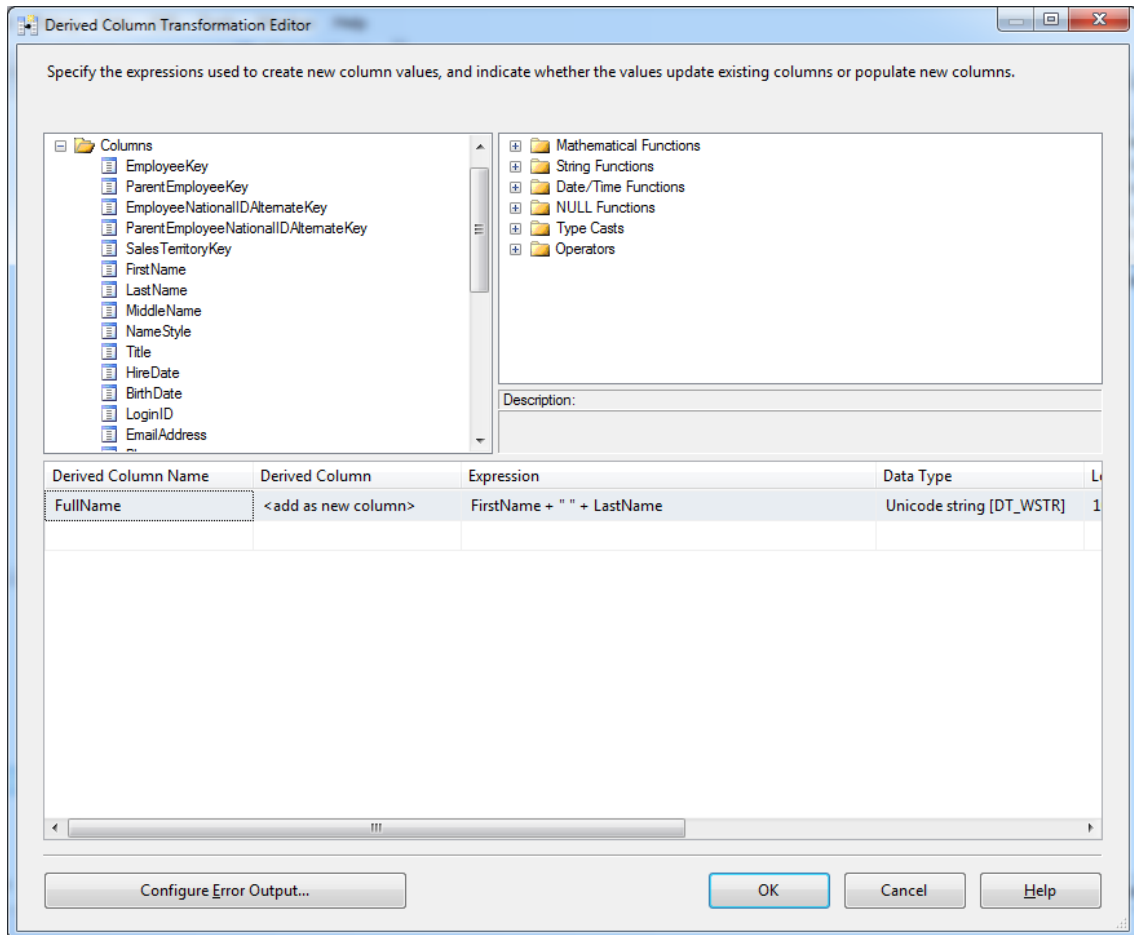


Figure 7–20. The Derived Column Transformation Editor dialog box allows you to create new columns in data flow using expressions that the data source can understand.

Next, drag the success data flow path from your Derived Column to your Sort. Open the Sort, and check `EmployeeKey` from the Available Input Columns, then click OK. Finally, connect Sort to OLE DB Destination. Your package should now resemble Figure 7–21.

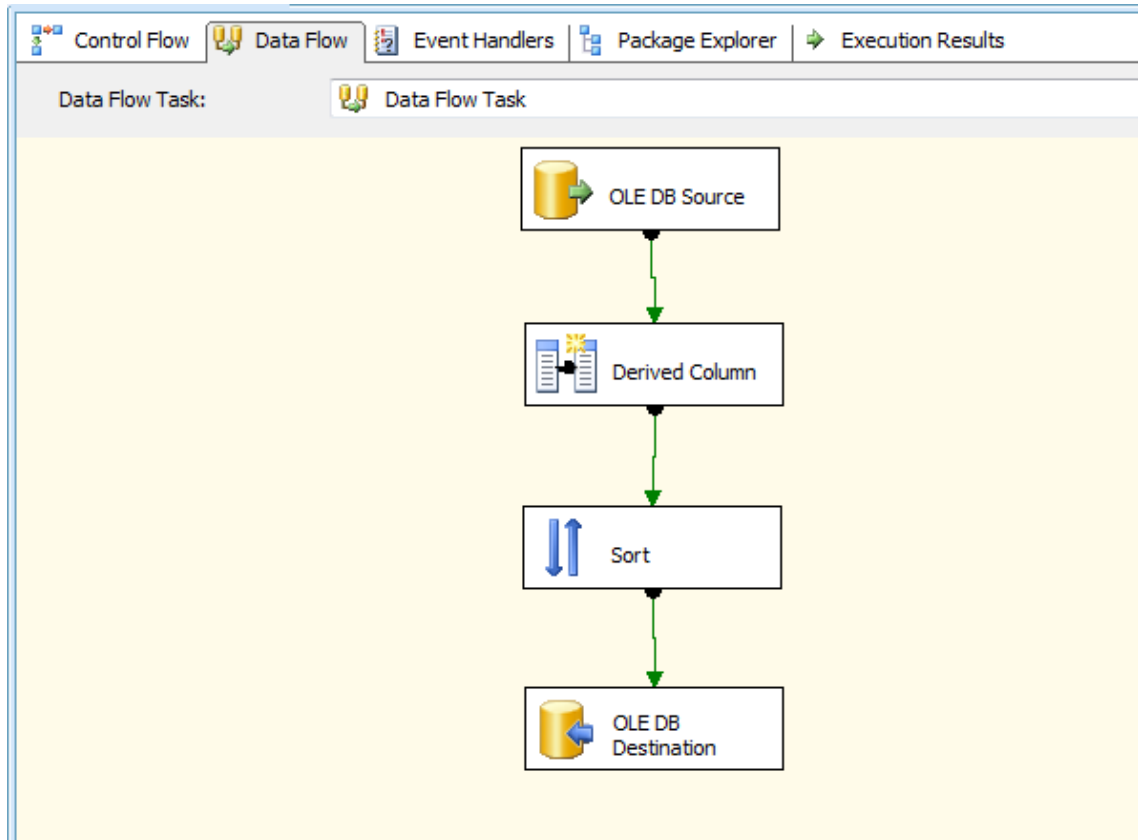


Figure 7–21. Adding some basic transforms to the data flow.

When you execute the package, the Progress tab becomes available in the SSIS designer. The Progress pane shows very detailed information about the execution of each task and step in the package. As shown in Figure 7–22, the level of detail can be very helpful in understanding the overhead involved with package execution.

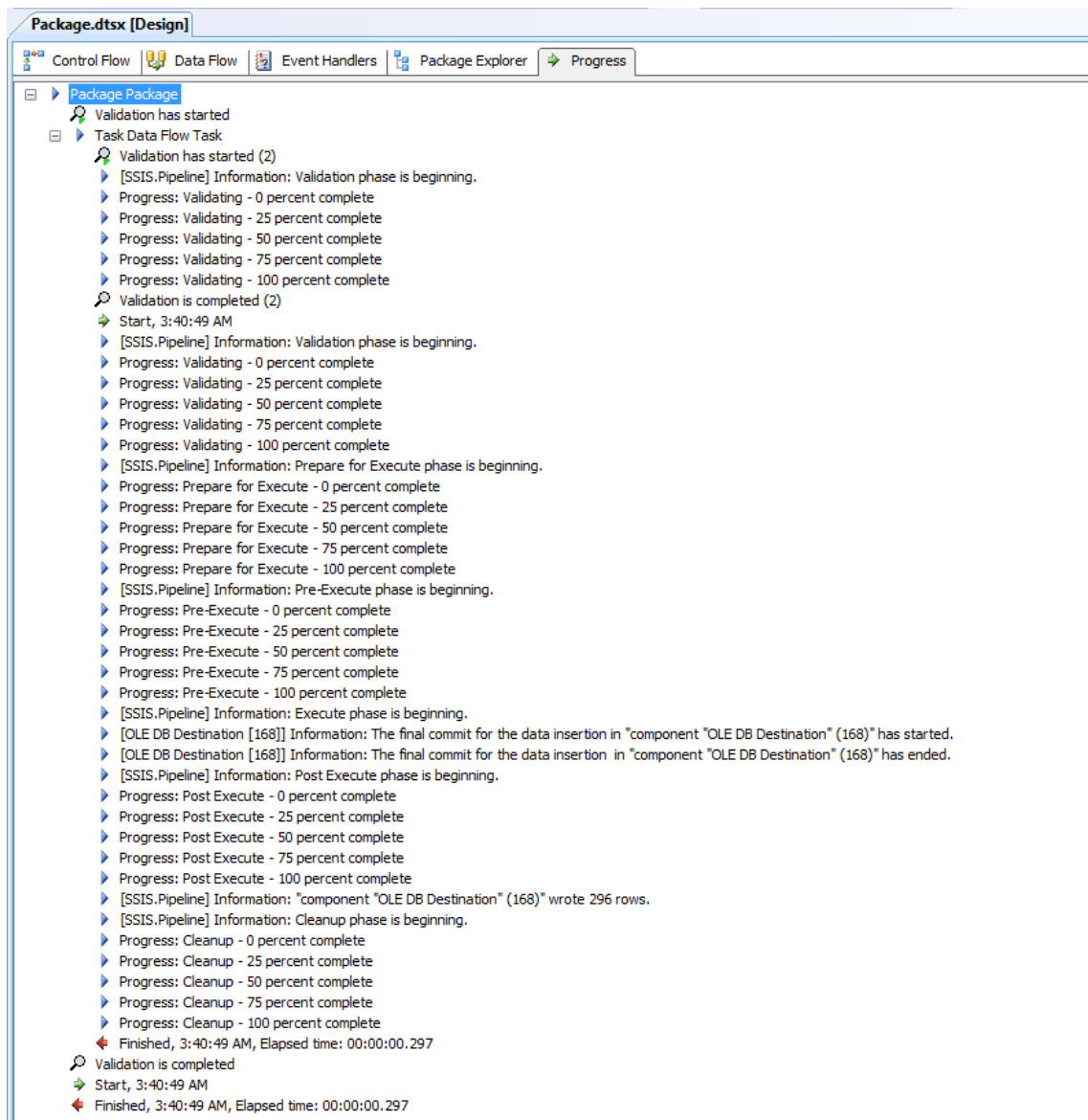


Figure 7–22. The Progress tab in the SSIS designer shows you detailed information about the overhead involved with executing each step and task of an SSIS package.

Summary

Are you beginning to understand the power of SSIS? This component of SQL Server is incredibly flexible and important to all BI solutions. We've covered only the basics of using the tools available (either in SSMS, with, the SQL Server Import and Export Wizard or in BIDS, with the SSIS Integration Services project type template) to create SSIS packages that will be the workhorses of your ETL processes for BI.

We've really just scratched the surface of what is possible with SSIS. In Chapters 8 and 9, we'll explore the data flow transformations and control flow tasks in greater depth. Also, you'll learn about error handling, debugging, and best practices for SSIS design in BI projects.



Intermediate SSIS

By now, you've translated your business requirements into an SSAS cube. The next task in implementing your BI project is to execute an ETL solution from your detailed data map. As discussed in Chapter 7, data maps are used as a basis for beginning ETL design. In our experience, a good deal of the initial work on a BI project consists of ETL process design, development, and debugging. We assume that you've completed your data map and created a simple package or two using the Import and Export Wizard or by using the SSIS package template in BIDS. In this chapter, we'll cover the following topics:

- Considering common ETL package-design practices
- Understanding control flow tasks
- Understanding data flow transformations
- Using the Dynamic Configuration Wizard and property expressions

Common ETL Package-Design Practices

Whether you start with the blank SSIS package project-type template in BIDS, or by using the Import and Export Wizard to create simple starter packages, your ETL solution may ultimately consist of many SSIS packages. Before you begin implementing any packages, you should take a bit of time to plan your ETL approach and high-level package design principles. The following list of common ETL practices is offered as a guide, based on our collective experience architecting and building enterprise BI systems: Create one set of packages for inserts and a different set for updates. Name the packages according to function.

- Create a minimum of one package for each dimension and fact table per action type (insert or update), for example, `insertCustomerDim` and `updateCustomerDim`.
- If you have a large number of data sources, consider creating one package per data source per dimension per action type. For example, one BI project we recently worked on had 16 data sources. Our naming convention was something like this: `insertCustomerDimFromCust1_xls` and `insertCustomerDimFromClient2_xls`.
- Use self-documenting naming conventions throughout each package. That is, apply common-sense names not only to package names but also to all items contained in packages (tasks, steps, and so on). Also, you can put notes on the design surfaces using the annotation capability in BIDS. This is a step that will pay for itself many times over; each time new ETL developers need to work on a package, they can read your notes, instead of e-mailing you or coming to your office to ask you questions. This is sounding better and better already, isn't it?

- Avoid hard-coded connection strings, user names/passwords, file paths, computer names, and so on in all packages using dynamic configuration (which will be explained in detail later in this chapter). This is critical to creating packages that can easily run in development, test, and production environments.
- If source data need significant cleansing, validation, and/or consolidation, use an interim staging database in your ETL process. Disk space is usually cheaper than additional RAM or CPUs.
- Break complex processing down into discrete steps for finer-grained control over performance and debugging. For very complex scenarios, use multiple packages, as simpler packages are easier to work with.
- Consider that some capabilities of SSIS are available in the Enterprise Edition of SQL Server only. We will note these features in this chapter's text as they are discussed.
- Consider that mastering SSIS's rich toolset takes a significant amount of time. Be careful when estimating your first ETL efforts. They will likely take longer than you expect.

Creating an SSIS Package from Scratch

In this chapter, we'll work either from scratch or from the samples available with SQL Server. As with designing your first cube, developing SSIS packages requires that you use BIDS to create a new project.

Creating the Package Itself

To get started, in BIDS, select a project of type Business Intelligence Projects via **File ► New ► Project** and then a template of Integration Services Project. Leave your project name as **Integration Services Project1**, pick a directory location, and then click **OK** to create the project. After you do that, you'll see the development environment shown in Figure 8–1.

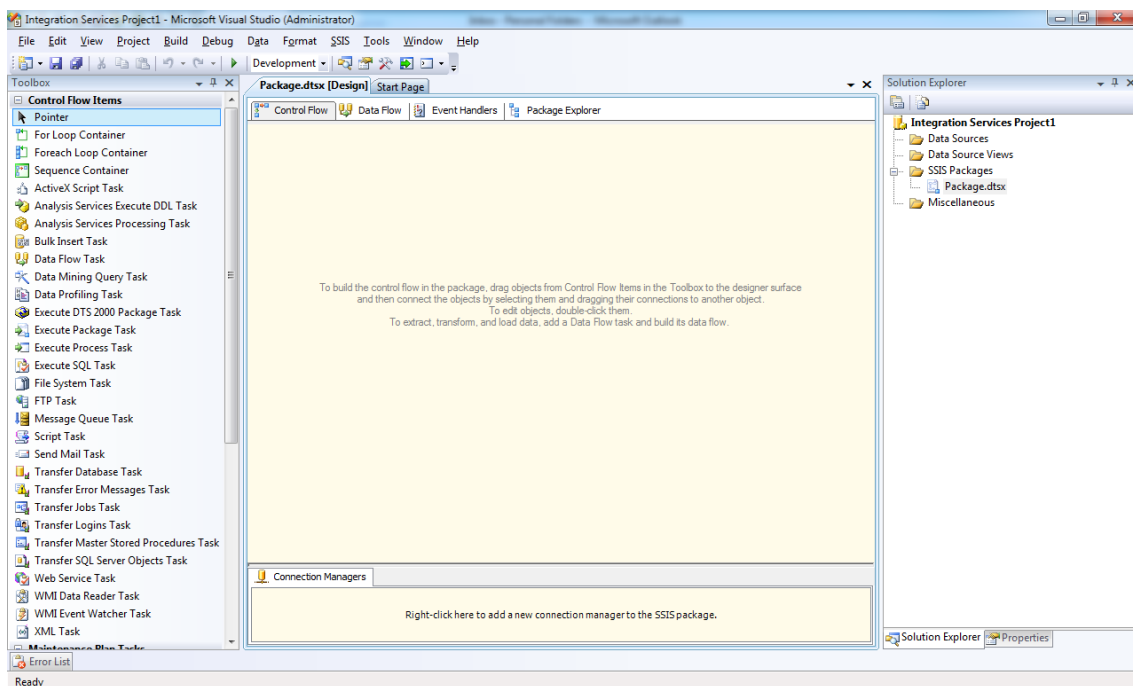


Figure 8–1. You author SSIS packages using BIDS with a project template type of *Integration Services Project*

We'll review each section of the designer now. First, to the left is the toolbox. The items displayed depend on which section you are working on in the SSIS package designer area (center). If you are working with the Control Flow, Event Handlers, or Package Explorer tabs, the toolbox will display the Control Flow Items; if you are working with the Data Flow tab, the Toolbox will display the Data Flow transformations. We'll be discussing both the control flow task and the data flow transformation types in greater detail later in this chapter.

Tip If you're feeling overwhelmed by all the different windows in BIDS and wondering how you're going to keep track of them all, relax! If you press **Ctrl+Tab** within BIDS, you'll get a list of all the open documents and a list of all the BIDS windows, and you can select the desired window by using the arrow keys. This technique should be a huge time-saver for you.

You may also choose to display variables associated with a package by clicking the SSIS item on the top menu bar and then clicking **Variables**. The Variables window will display in the same location as the toolbox by default, as shown in Figure 8–2. In this section, you can view and add variables to the package as well. It is also important to remember that variable names are case-sensitive in SSIS.

In Figure 8–2, the icon next to the last variable (TestVar) is blue to indicate that the variable is user defined. One common source of frustration for new SSIS developers is variable scope. Variables with a scope of Package are visible to all data flows, and other containers. The Scope column can prove quite helpful when debugging your SSIS packages. The other variables that are listed in Figure 8–2 are created automatically when you create a new SSIS package using that template in BIDS, and they are colored gray.

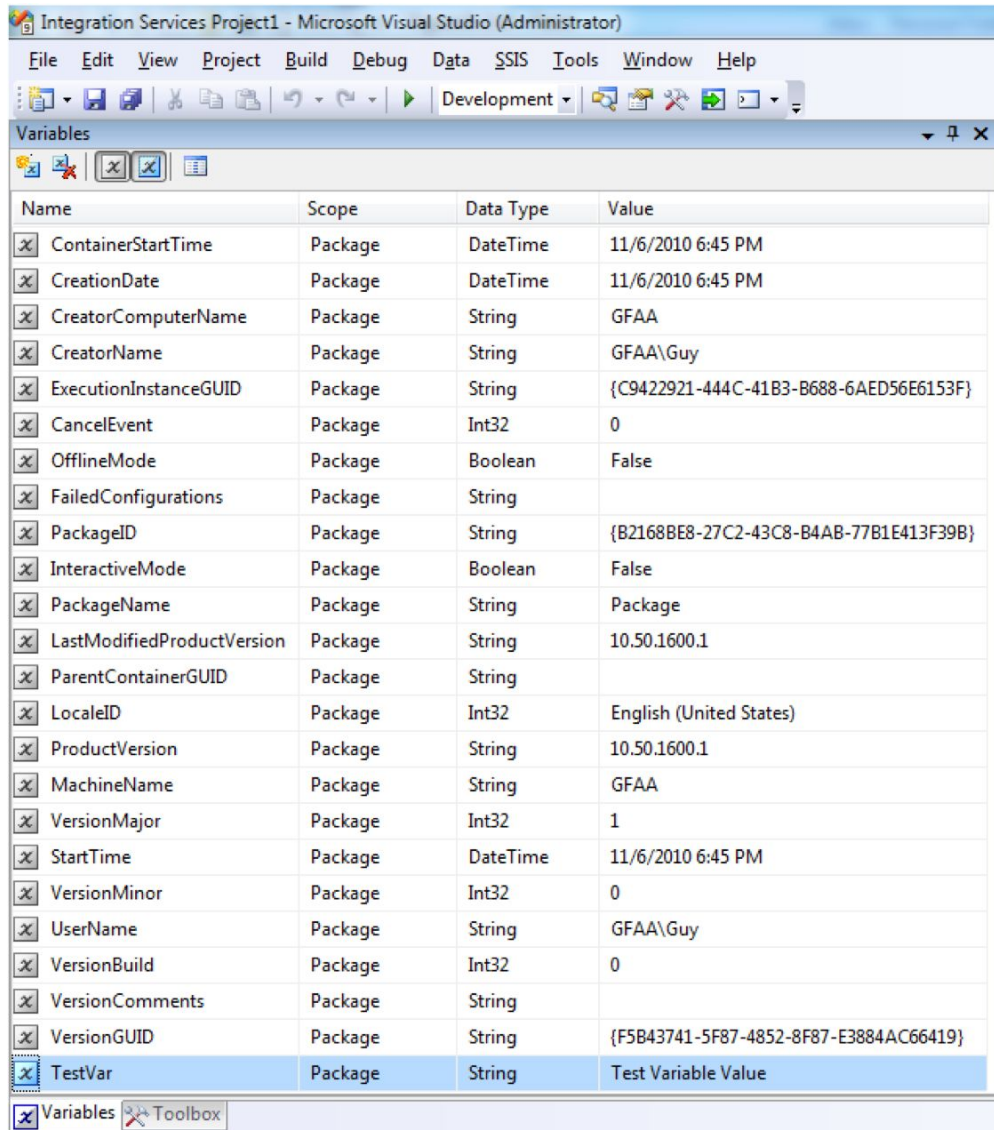


Figure 8–2. You can view variables associated with SSIS packages in BIDS by clicking the SSIS menu and then clicking Variables. You can also add variables via this interface.

When working with the Control Flow, Data Flow, or Event Handlers tabs, the BIDS designer will display the Connection Managers work area at the bottom of the screen. The Connection Managers area displays an icon for each connection used in the package. You can add new connections by right-clicking in this area, selecting the connection type from the menu, and configuring the connection.

On the right side of the BIDS SSIS package designer interface is Solution Explorer. When you open the SSIS template, you are actually working with a Visual Studio project template. SSIS projects in Visual Studio are containers for multiple files, in this case, multiple .dtsx files or packages and related files, like data sources (or connections) and data source views (DSV). Projects allow the information in them to be shared between files in the same project. In the case of SSIS, the most common items to be shared are represented as folders in the Solution Explorer tree.

On the bottom right (by default) of the BIDS interface is the properties area. As you saw with SSAS projects, you can select a particular item from the drop-down list in the property area, or you can simply click the item you want to set the properties for on the design surface. After you've selected the appropriate item, the properties for that item will appear. Read-only properties are greyed out; the other properties are configurable.

The last part of the SSIS interface is the SSIS menu, which contains the following items:

- *Logging*: This item allows you to create package logs in up to five different destination types (such as file, SQL Server, and so on). These logs capture runtime events that are enabled for logging. This option is covered in more detail in Chapter 9, “Advanced SSIS.”
- *Package Configurations*: This starts the Package Configuration Wizard. This wizard allows you to dynamically configure various package properties. The implementation of this wizard is covered in more detail in the “Using the Dynamic Package Configuration Wizard” section of this chapter.
- *Digital Signing*: This option allows you to associate a certificate with your SSIS package. Signing a package will allow you to prevent it from running if its contents have been altered.
- *Variables*: With this option, you can display the package variables window on top of the toolbox by default. It allows you to view package variables and add variables as needed.
- *Work Offline*: This one allows you to work in a disconnected fashion. The default is to work in connected mode so that connections to data sources and DSVs can be used to retrieve metadata (for example, source table column names and data types) that is used to populate SSIS tasks.
- *Log Events*: Use this option to display the Log Events window over the top of the toolbox by default, as shown in Figure 8–3.

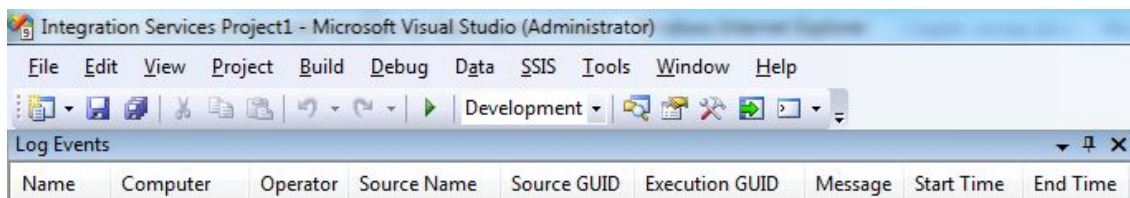


Figure 8–3. The Log Events window is displayed in BIDS after you open it from the SSIS menu. It shows information about events you've chosen to capture for a particular SSIS package.

- **New Connection:** This one allows you to create a new connection for use in the package. These connections can be based on data sources (global to the project) or can be local to the specific package that you are working on. All connections will appear in the Connection Mangers window at the bottom center of the design area after you add and configure them.
- **View:** The final option allows you to switch between tabs in the SSIS designer, that is, Control Flow, Data Flow, and so on.

When you execute a package in BIDS, by clicking the green triangle on the toolbar or by right-clicking the .dtsx file in Solution Explorer and then clicking **Execute Package**, an additional tab becomes available in the BIDS designer. The new tab is called **Progress** and will show you step-by-step execution results. A sample is shown in Figure 8–4.

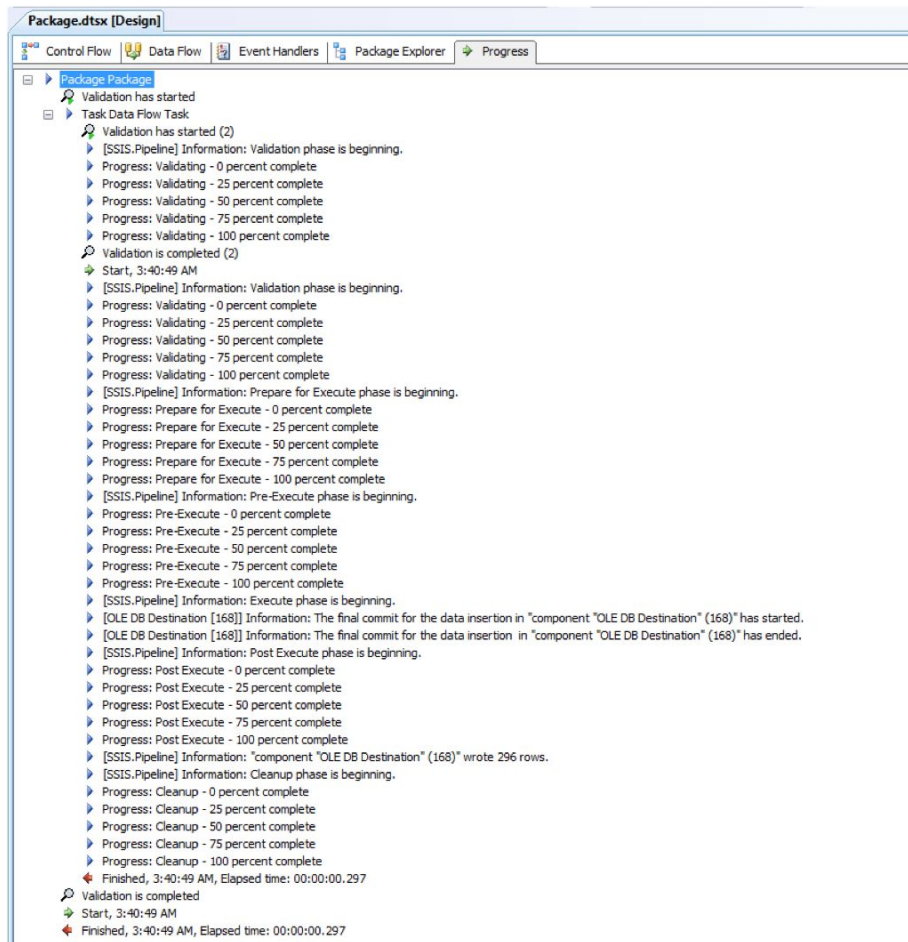


Figure 8–4. The **Progress** tab is displayed in BIDS after you execute an SSIS package. It shows detailed execution information for the package.

Now that we've reviewed the BIDS design environment, you're ready to create your first SSIS package by designing it from scratch, rather than by running the Import and Export Wizard as we did in Chapter 7. To understand what is now possible with SSIS, you should consider an SSIS package to be an executable file that will perform a data workflow. You can combine the traditional ETL data for BI projects with many other types of tasks to create the type of customized data workflow that best suits your particular business needs.

Another important point to consider is that neither the data sources nor the data destinations need to be any version of SQL Server. SSIS can connect to anything for which there is a supplied provider. There are many provider types supplied with SSIS, and the list continues to grow with each release. With all this power, you may be wondering where to start. All data workflows need to connect to something, so the logical starting point is configuring connections.

Configuring Connections

Although it is possible to define connections only within each package, as a matter of convenience, you will most often find yourself reusing connections across multiple packages. To create a new connection in this way, right-click the Data Sources folder in the Solution Explorer window for your project, and choose New Data Source. After you do this, click Next, and you'll see a Data Source Wizard dialog box in which you can either reuse a previously defined connection or create a new one. The "Create a data source based on an existing or new connection" option in the "Select how to define the connection" dialog box is shown in Figure 8-5.

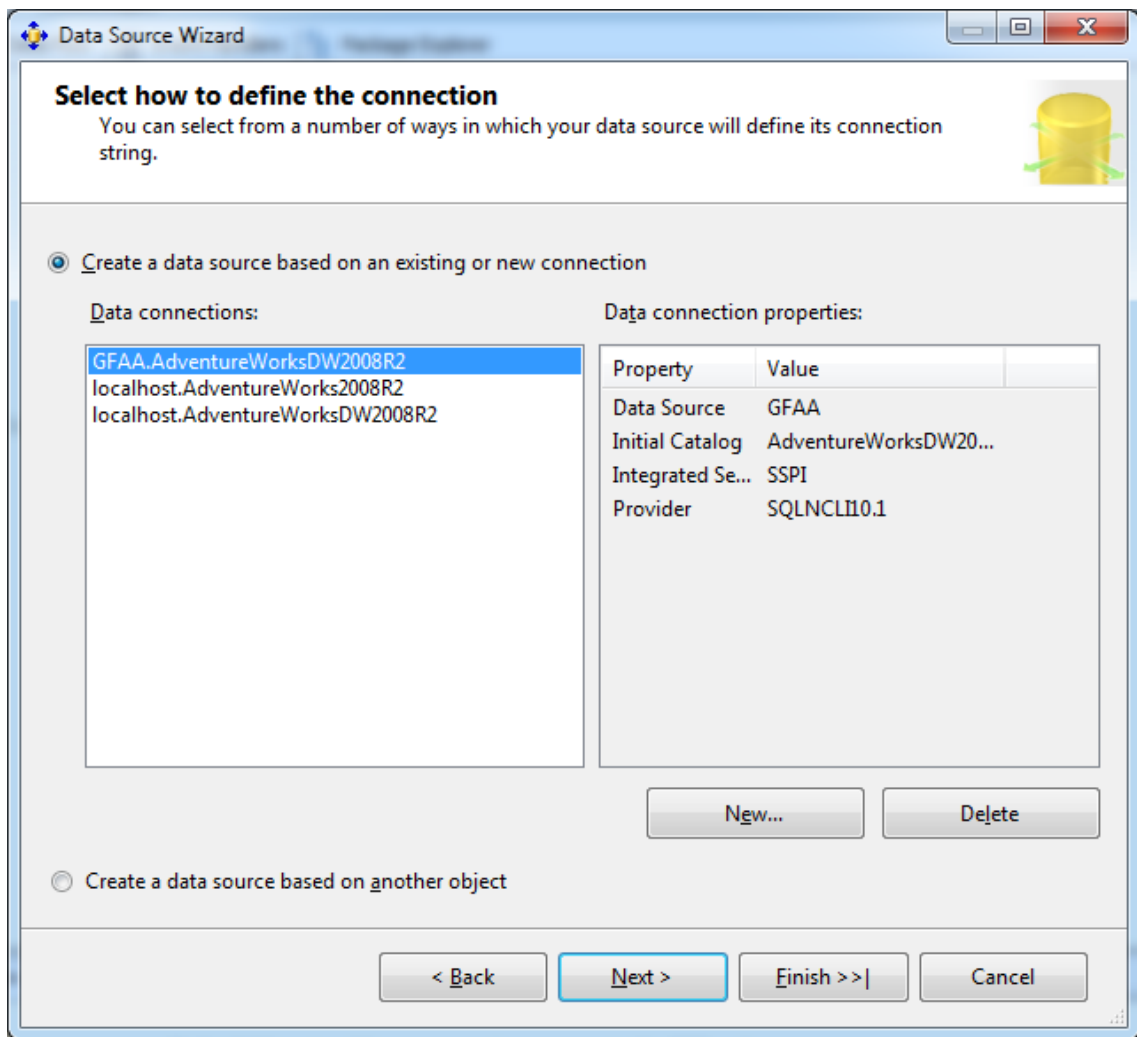


Figure 8–5. In the “Select how to define the connection” dialog box in BIDS, you configure reusable connection based on existing connections, other objects, or new information.

The next step is to associate this global connection with the specific package that you are working on. To do this, right-click the bottom center Connection Managers pane, click New Connection from Data Source, and select the connection name that you previously created. To review the broad variety of data source types that can be used in SSIS packages, right-click inside the Connection Managers pane, and choose New Connection. Figure 8–6 shows the list of Connection Manager types that are available to you.

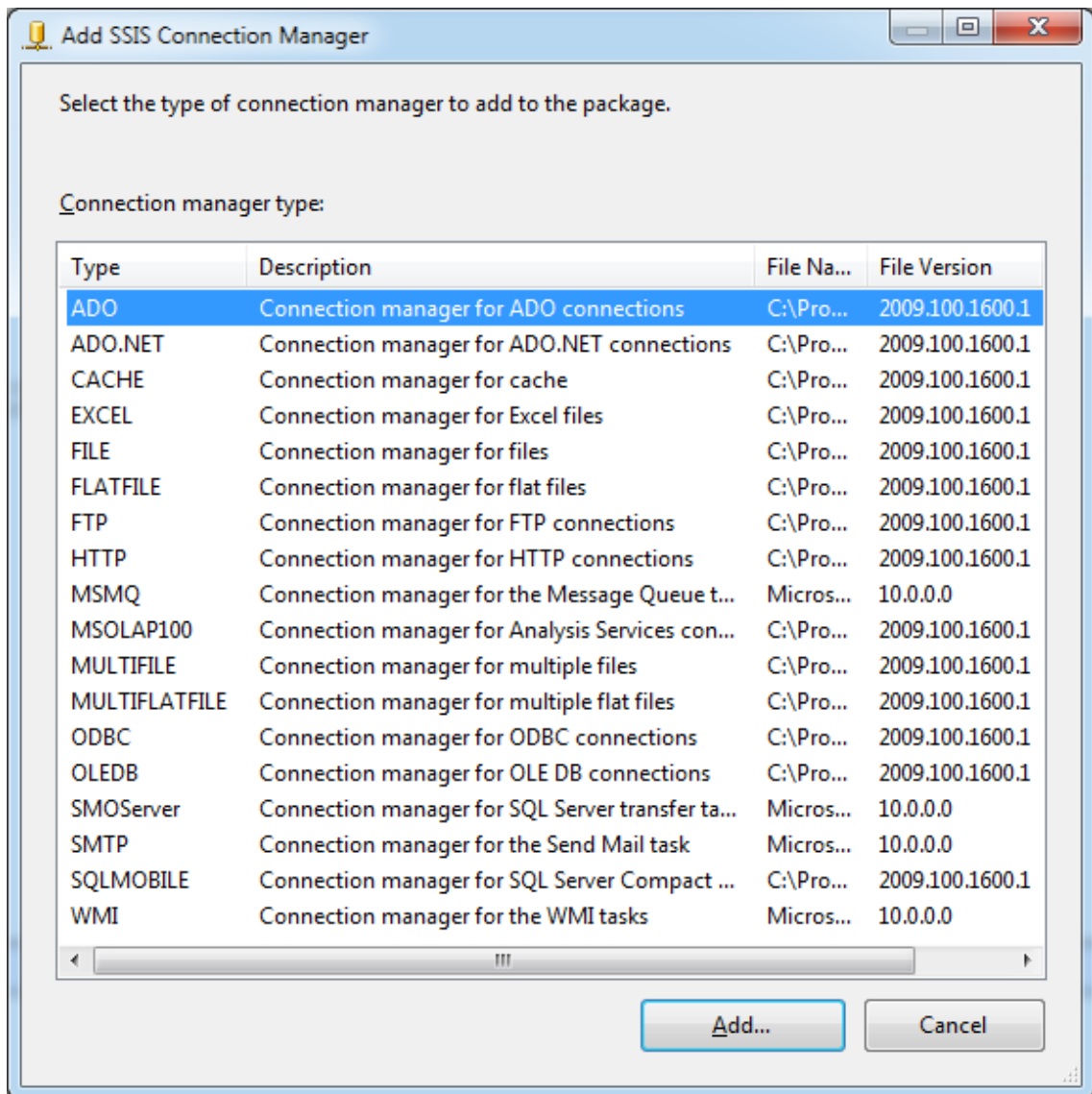


Figure 8–6. When you right-click in the Connection Managers pane, and select New Connection, you are presented with a wide variety of data source types to select from.

Interestingly, different types of connections are available when you right-click the Connection Managers area of the SSIS design surface versus those that are available if you right-click the Data Sources folder in the Solution Explorer list. Figure 8–6 shows the connection types available for you from the former area. Generally these include more file-based types, such as FLATFILE, EXCEL, MULTIFLATFILE, and so on.

The top-level types of connections available for you from the latter area are .NET Providers, .NET Providers for OLE DB, and Native OLE DB providers. These types are more relational in nature, that is, they're intended for use with SQL, Oracle, and so on. You can see the list of specific providers when you expand each of these three folder types; for example, the .NET Providers include SqlClient Data Provider, OracleClient Data Provider, and Odbc Data Provider.

Using Data Source Views (DSVs)

Another type of component shared across packages is a data source view (DSV). DSVs in the SSIS designer function identically to the way they work in the SSAS designer. As with SSAS, you use and customize DSVs when you do not have permissions to create views or other types of abstractions, such as calculated columns, in the underlying data source. An option available to you when creating DSVs is the ability to restrict the view to a particular schema in the source database. This option supports the capability in SQL Server to group objects in a relational database together by schema. If you are not familiar with the concept, you can think of schemas like folders on the file system. The Advanced Data Source View Options dialog box is shown in Figure 8-7.

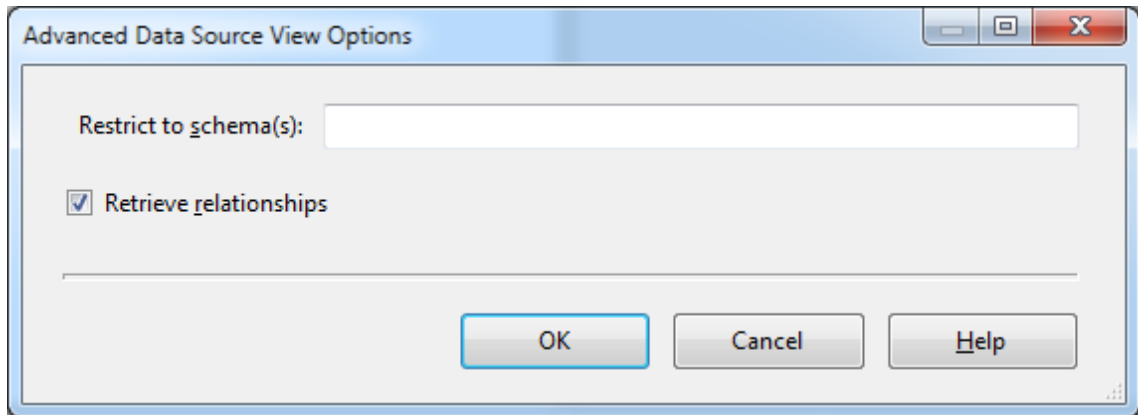


Figure 8-7. When you create a DSV, you can restrict the view to a particular schema in the source database via this dialog box.

Another useful aspect of using DSVs while working in BIDS is the ability to explore the data contained in the view. To do this, you just right-click any table in the DSV work area. As was mentioned in Chapter 3, you can then view a subset of the data as a table, chart, pivot table, or pivot chart. This can help you better understand the type and quality of the source data before you start creating your SSIS package. Again, you'll find this capability most helpful when you are not permitted to directly query the source data.

Although using shared objects can be useful for your ETL solution development, you do not have to use shared data sources or DSVs to create an SSIS package. This differs from the process of creating a SSAS cube, where both a shared data source and a shared DSV are required.

Reviewing the Included Samples Packages

Now that we've completed our tour of the BIDS environment and the common areas, we'll start working through the sample packages. These packages are located by default at C:\Program Files\Microsoft SQL Server\100\Samples\Integration Services\Package Samples. We'll use these packages as a platform to build our understanding of two of the core parts of the SSIS design interface: using the control flow tasks and data flow transformation.

The first package we'll start with is the Execute SQL Statements in Loop sample. To open this package in BIDS, navigate to the ExecuteSQLStatementsInLoop Sample folder and double-click the ExecuteSQLStatementsInLoop.sln file. This opens the SSIS solution in BIDS. Next, double-click ExecuteSQLStatementsInLoop.dtsx under the SSIS Packages node in Solution Explorer. This opens the package to the Control Flow design surface and is shown in Figure 8–8.

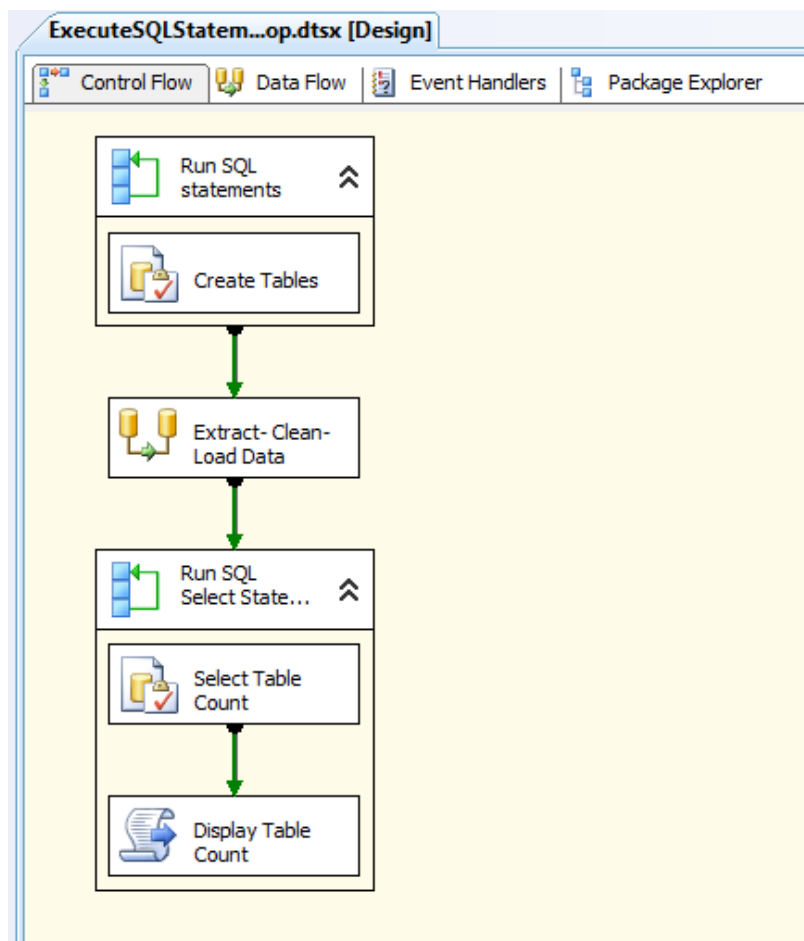


Figure 8–8. The first sample package we are going to work with is called *Execute SQL Statements in Loop*. This package is a good example of using the control flow *Foreach* loop container.

Adding Control Flow Tasks

Your next step in package design will be to select one or more control flow items from the toolbox and drag them to the Control Flow design surface section. Figure 8–9 shows the task types available via the toolbox for the control flow items.

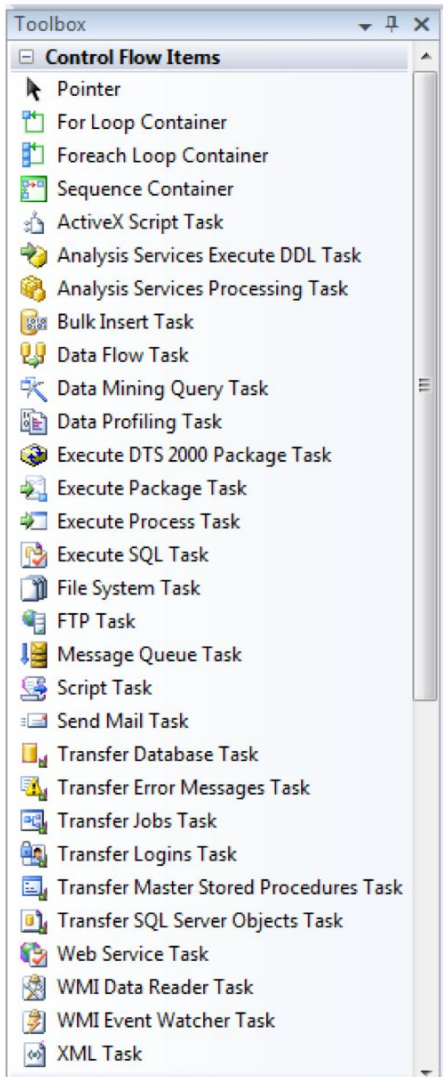


Figure 8–9. The Control Flow Items Toolbox in BIDS allows you to add different types of control flow tasks to your SSIS package.

To help you best understand all of these tasks, we'll group them into types. We'll also refer to our sample package for this discussion. The task type groups for the control flow tasks are as follows:

- *Containers*: Allows you to group one or more tasks inside of it. These include the For loop, the Foreach loop, and the Sequence containers. There is also a container object called TaskHost that is designed to encapsulate a single task. The TaskHost object is not visible in the Control Flow toolbox and its properties are set by the task it contains. For more information, see the BOL topics "TaskHost Container" and "TaskHost Class." You also have the ability to group tasks together. This grouping of tasks places grouped tasks inside of a container-like Group object.
- *SQL Server*: Allows you to work with SQL Server objects, including the Bulk Insert, Execute SQL, Transfer Database, Transfer Error Message, Transfer Jobs, Transfer Logins, Transfer Master Stored Procedures, and Transfer SQL Server objects tasks.
- *Data Preparation*: Allows you to work with the local file system, including the File System, FTP, Web Service, Data Profiling, and XML tasks.
- *Workflow Tasks*: Allows you to execute other processes, including Execute Package, Message Queue, WMI Data Reader, WMI Event Watcher, and Send Mail tasks.
- *Script*: Allows you to execute some type of script, which include the ActiveX Script and Script tasks. ActiveX Script tasks are deprecated and are included for backward compatibility with DTS.
- *SSAS*: Allows you to perform some sort of processing on SSAS objects, including the SSAS Execute DDL, SSAS Processing, and Data Mining query tasks.

You can also extend the set of control flow tasks or data flow transformations available in BIDS by downloading, purchasing, or developing your own SSIS tasks and then installing them into the BIDS environment. You install new components by right-clicking the toolbox, and then clicking Choose Items. This brings up the Choose Toolbox Items dialog box, which allows you to add a toolbox item, as shown in Figure 8–10.

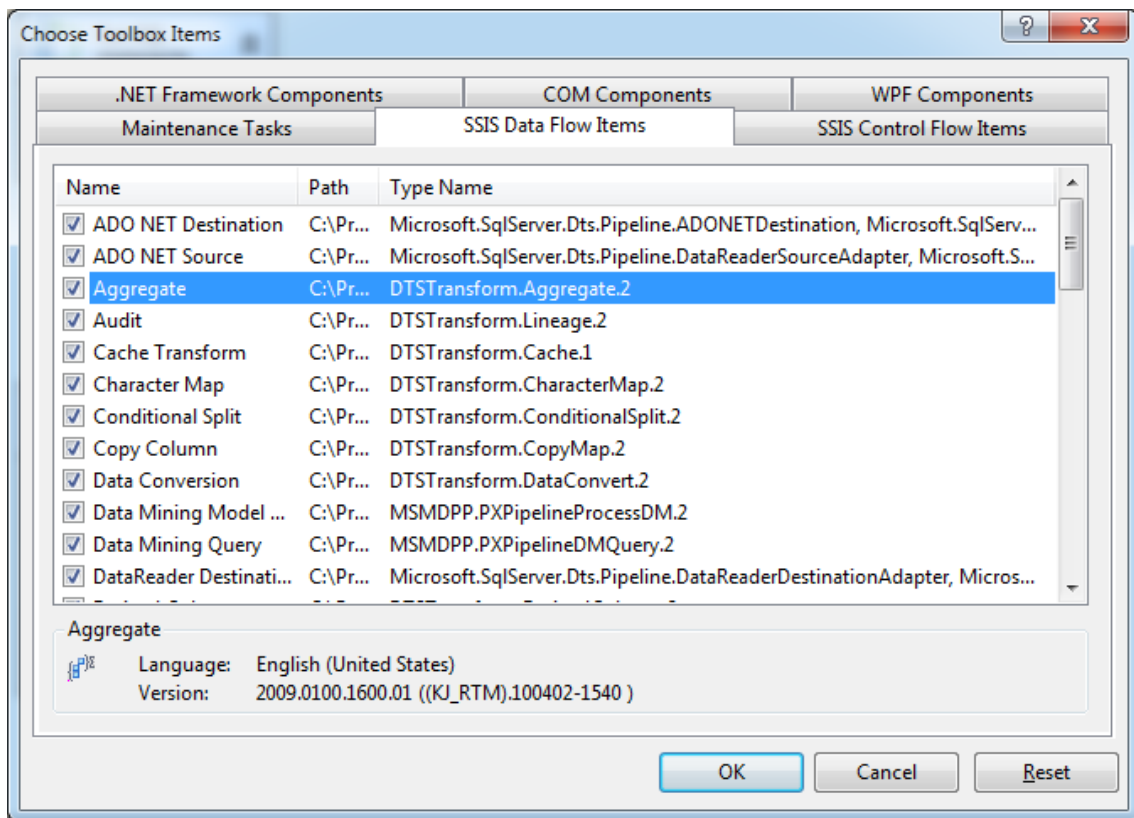


Figure 8-10. You can install new components that you download, purchase, or create into the BIDS SSIS control flow or data flow transformations by right-clicking the Toolbox and then clicking Choose Items.

Now that we've grouped the control flow tasks by type, let's dig in a bit deeper to the capabilities of the tasks most frequently used in BI ETL solutions.

Container Tasks

Container tasks allow you to group and iterate over contained tasks. They include the For loop, Foreach loop, and Sequence tasks in the Control Flow toolbox. You also have the option to simply group multiple child tasks together by selecting the tasks on the design surface, right-clicking, and then choosing Group. This type of an object looks like an SSIS container, and shows up on the SSIS design surface as a collapsible rectangle with other tasks inside of, however, it is really more like a code region in Visual Studio in that it allows you to collapse complex sections of packages, thereby making them more human-readable.

To use a Foreach container task, you must configure the properties of the enumerator by right-clicking the task on the design surface, and choosing Edit. You can select from these choices of Foreach enumerators: File, Item, ADO, ADO.NET, From Variable, NodeList, and SMO. When you do so, you'll see the Foreach Loop Editor dialog box shown in Figure 8-11.

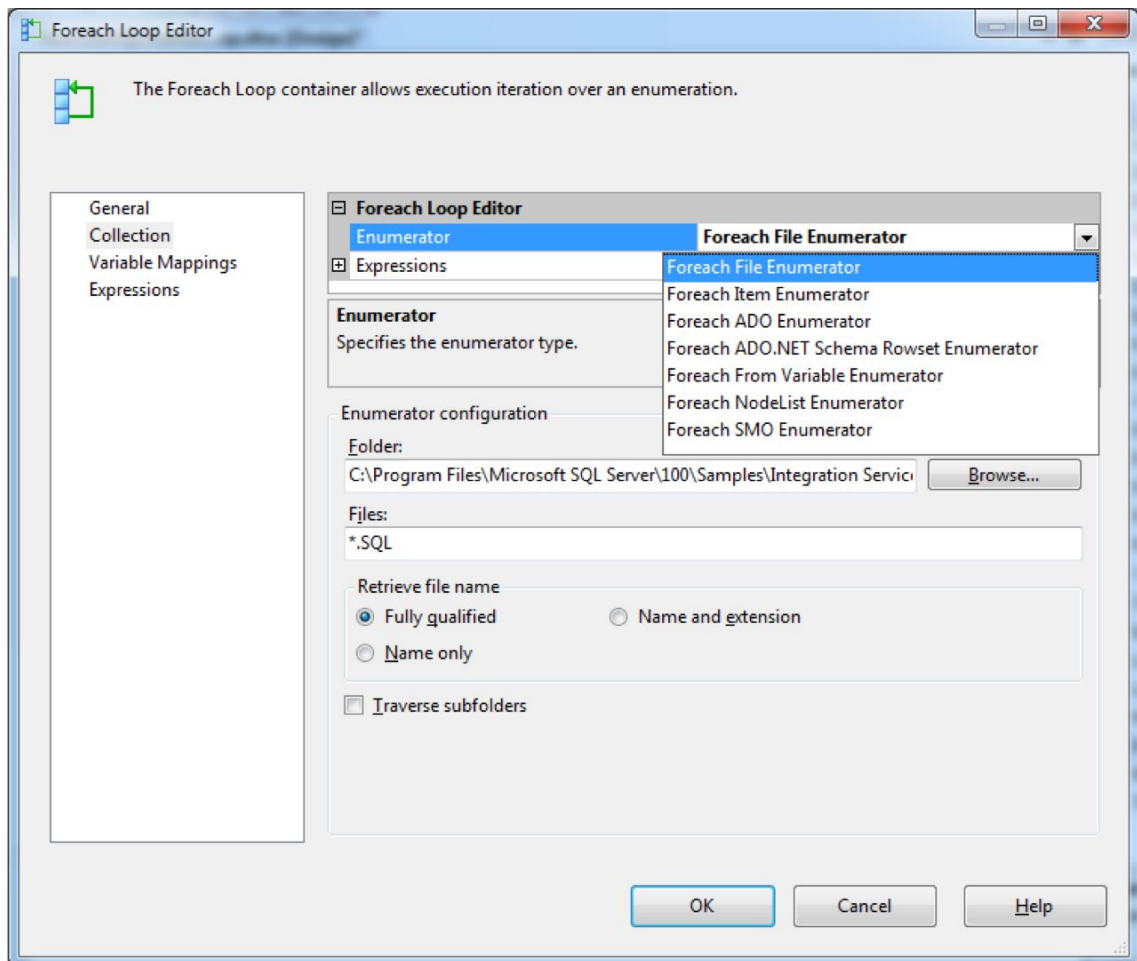


Figure 8–11. Using the Foreach loop container task requires that you configure the enumerator.

■ **Note** SMO (SQL Management Objects) is the management object model used in SQL Server.

SQL Tasks

The next group of tasks allows your SSIS package to execute a SQL statement. The most commonly used is the Execute SQL task. In our sample package, shown earlier in Figure 8–8, the execute SQL task will create tables based on information contained in a SQL script file.

The script file is configured in the property sheet for this task as shown in Figure 8–12. Note also that this task will iterate because in the example package, it's nested in a Foreach loop container. Another

consideration with Execute SQL tasks is the ability to associate parameters with a query. This is configured in the same dialog box.

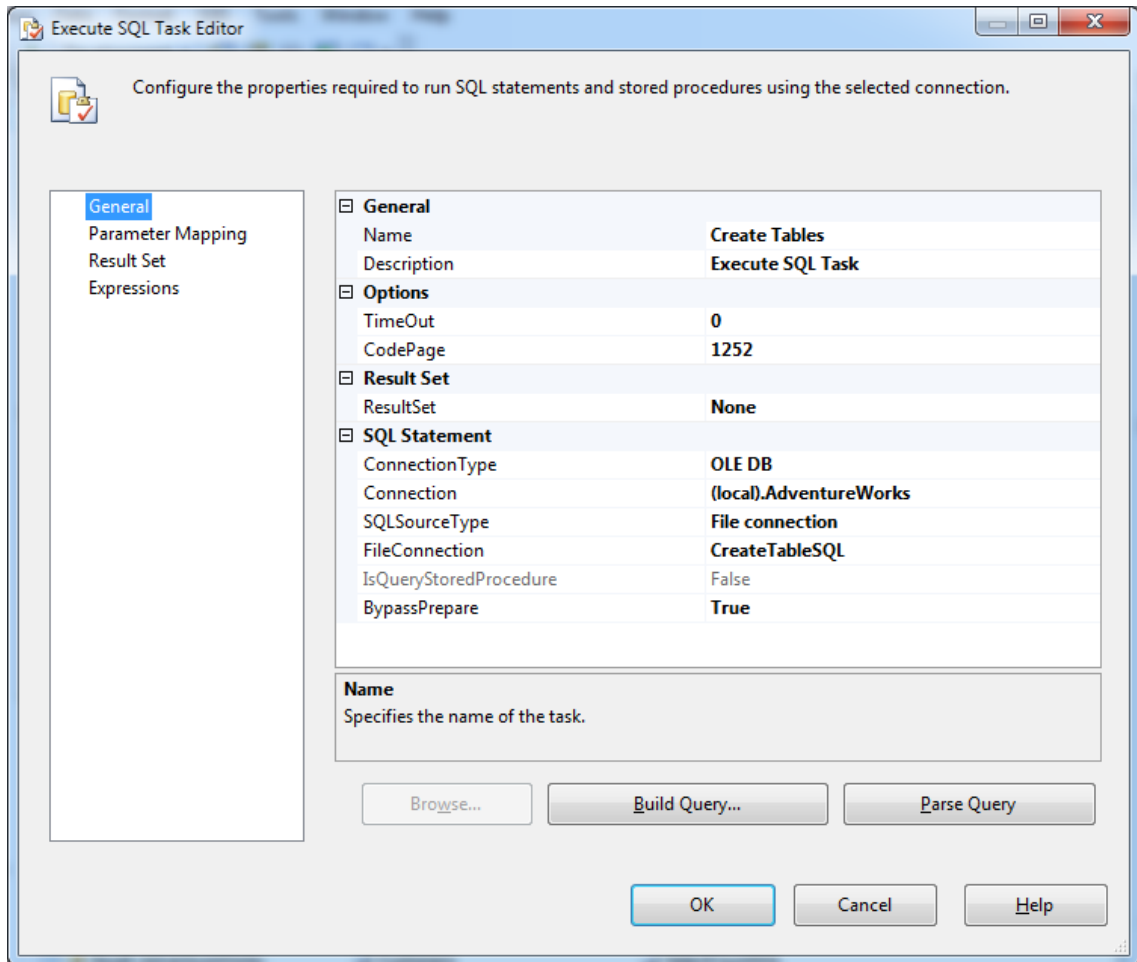


Figure 8–12. In the configuration dialog box for the Execute SQL task, you can associate the SQL query with a connection to a file (which tells the task to get the SQL to execute from the file contents) by configuring the *SQLSourceType* property to “File connection”.

The next most common type of SQL control flow task used in ETL solutions is the data flow task. The Data Flow task is shown as a task rectangle on the Control Flow area in Figure 8–8 (shown previously); for our example, this has been named Extract – Clean – Load Data. In SSIS, the data flow task detail is exposed via the Data Flow window. To get there, double-click the Data Flow task on the Control Flow design surface. You’ll then be taken to the Data Flow work area for that particular data flow task, as shown in Figure 8–13.

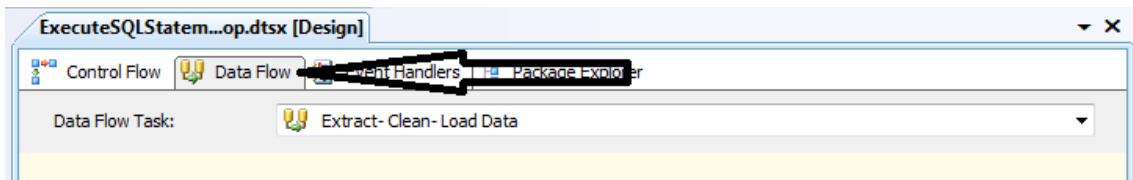


Figure 8-13. The Data Flow work area in BIDS allows you to visually configure a data flow for a particular data flow task from the control flow area of your SSIS package.

Once in the Data Flow area, you can configure the data flow by using source and destination components, as well as the transformation components available in the Data Flow toolbox. You'll learn more about the procedures for doing this in the "Understanding Data Flow Transformations" section later in this chapter. For now, however, you may be interested in what the Data Flow area looks like for our particular sample package; Figure 8-14 shows the Data Flow area.

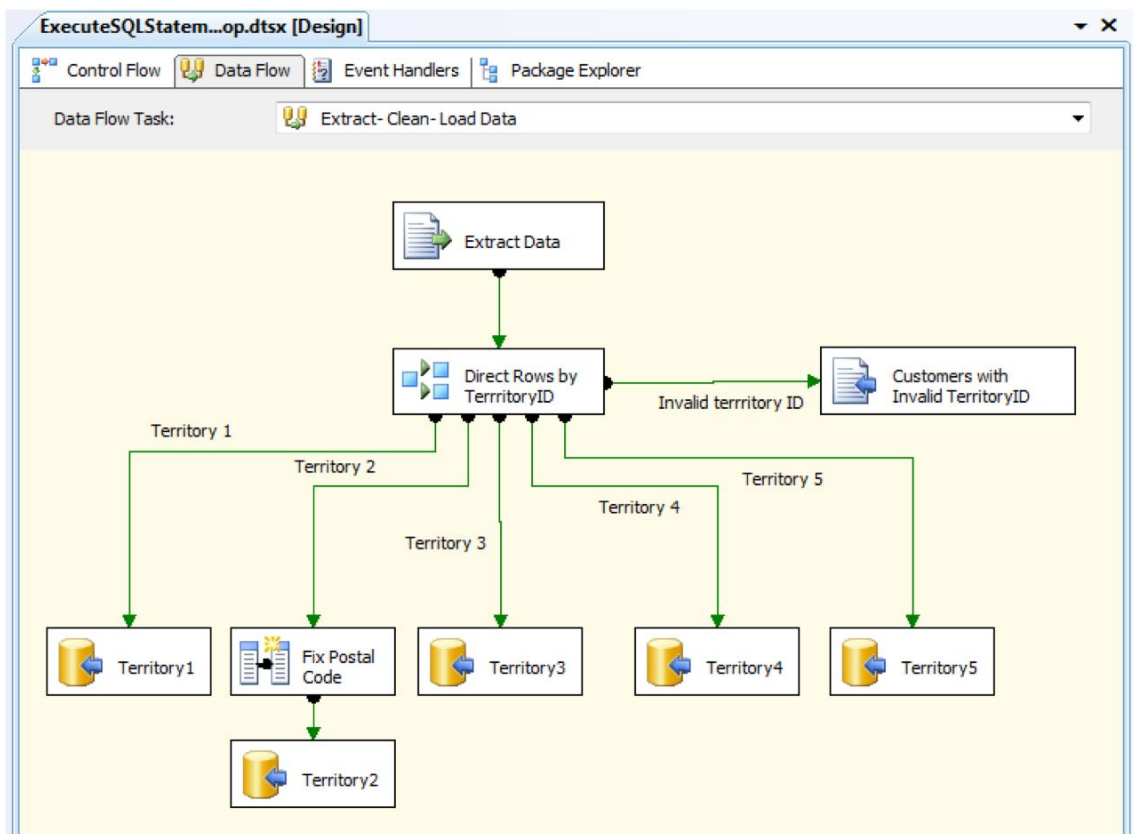


Figure 8-14. The Data Flow work area for the ExecuteSQLStatementsInLoop sample package shown in Figure 8-8.

Another commonly used SQL/file system control flow task is Bulk Insert. We've listed this task in both categories because it executes a SQL `BULK INSERT` statement to move text data from the file system into a SQL Server table.

File System Tasks

The XML task supports the industry direction to move away from .txt or .csv files, towards structured .xml files. The samples include a `ProcessXMLData.dtsx` package that shows the use of this task. To open this package, navigate to the sample files, and open the .sln file with the same name. Open the package in BIDS by double-clicking it, and you'll see two XML tasks on the control flow design surface.

To configure the XML task, right-click it, choose Edit, and select the type of operation you want to perform. The selection you make here changes the dialog box to add/remove additional configurable supporting properties. Here are your choices:

- *Validate*: Allows you to specify that the package should validate an XML document against an XML schema document (XSD) or XML document type definition (DTD).
- *XSLT*: Allows you to specify that the package should perform an XSLT transformation against the configured XML document.
- *XPath*: Allows you to specify that the package should run an XPath query and then do some sort of evaluation against the named section of the configured XML document.
- *Merge*: Allows you to specify that the package should merge two XML documents.
- *Diff*: Allows you to specify that the package should compare two XML documents and return some result based on the comparison, that is, fail on difference, return a DiffGram, and so on. A DiffGram is an XML format that identifies current and original versions of data elements.
- *Patch*: Allows you to specify that the package should apply a DiffGram to a configured XML document, with the result being a combined, new XML document.

Figure 8–15 shows the configuration dialog box for the XML task with the XPath operation type selected.

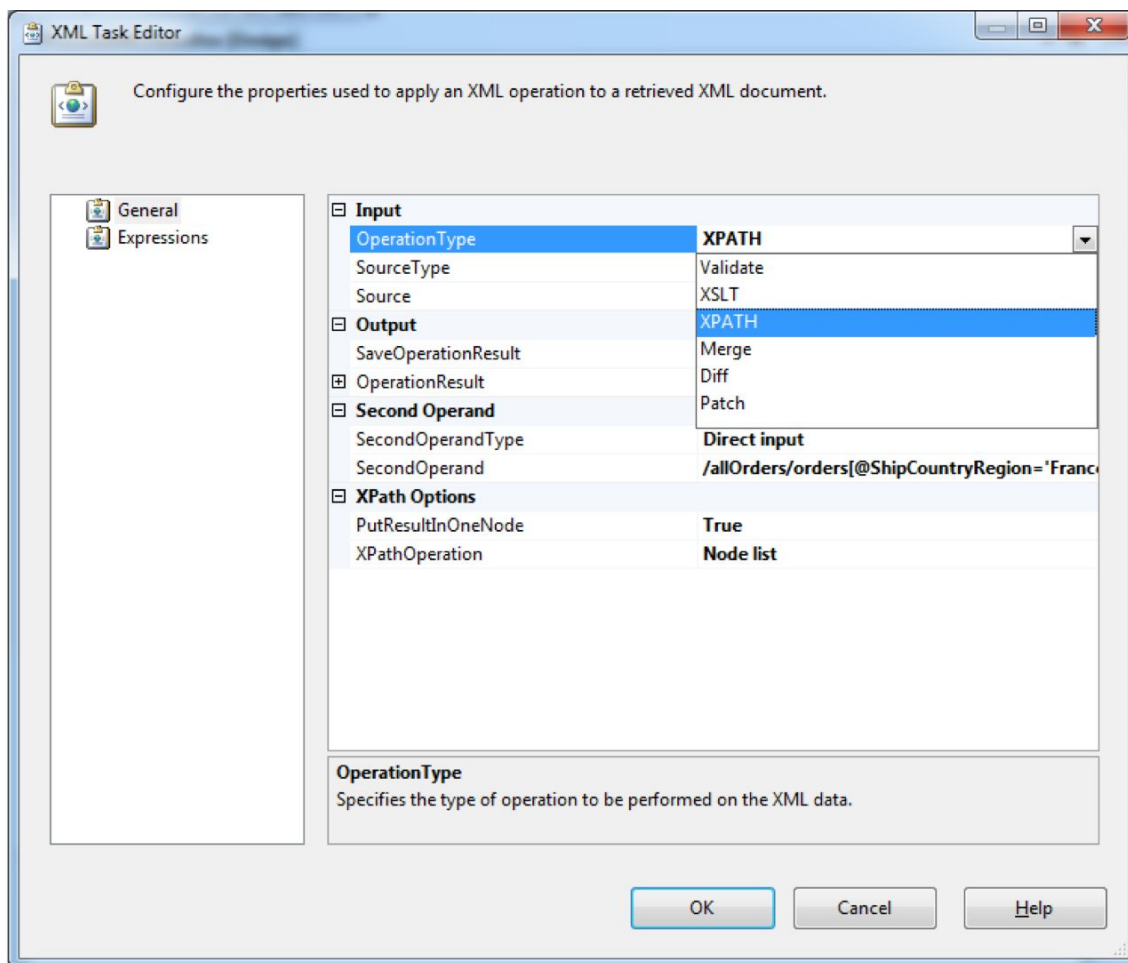


Figure 8–15. The XML task has different types of XML operations for you to select from when configuring this task. This example shows the configuration values for the XPath option.

Operating System Tasks

This type of task is where you will find the two Windows Management Instrumentation (WMI) tasks: WMI Data Reader and the WMI Event Watcher. Both of these tasks allow you to add the ability for your SSIS package to listen for values associated with the operating system—for example, hot fixes installed or events that have fired—and then continue on with the control flow of the SSIS package based on the results of this listening. WMI uses a query language that is similar to T-SQL, which is called the WMI Query Language (WQL).

To use either of these tasks, you must add a connection of type WMI to your package. Then you associate that connection with whichever type of WMI task you are working with. To configure the

properties of a WMI task, you right-click it. The most important property is the WMI Query. BOL has several examples.

■ **Tip** If you do use the WMI tasks, there are some tools that can help you to quickly author WQL queries. A freely downloadable WMI query builder is available at

<http://www.microsoft.com/downloads/details.aspx?FamilyID=2cc30a64-ea15-4661-8da4-55bbc145c30e&DisplayLang=en>.

The Message Queue (MSMQ) task is also available in this category. It allows you to configure your SSIS package to send or receive particular messages in a Microsoft Message Queue.

Script Tasks

If you choose to add any script to the control flow area of your package, you now have two choices regarding the language. The ActiveX Script task, which uses VBScript, is still available. As mentioned previously, be aware that the ActiveX Script task is included for backward compatibility and will be removed in a future version of SQL Server. Your second choice is the ability to add scripts using Visual Basic 2008 or Visual C# 2008. To do this, you add a Script task to the Control Flow area. You then configure the task by right-clicking the icon on the design surface, and choosing Edit. In the Script Task Editor dialog box, click the Script option on the left, and then click the Edit Script button on the bottom right. This opens the Microsoft Visual Studio 2008 Tools for Applications (VSTA) scripting window. Here, you can write your script with full IntelliSense. You also can set breakpoints to assist you with script debugging. You do this by clicking in the margin (grey area) to the left of the line of script where you want to break. A sample script is shown in Figure 8–16, using the sample package Sync Partitions.dtsx, which is in the SyncAdvWorksPartitions sample solution.

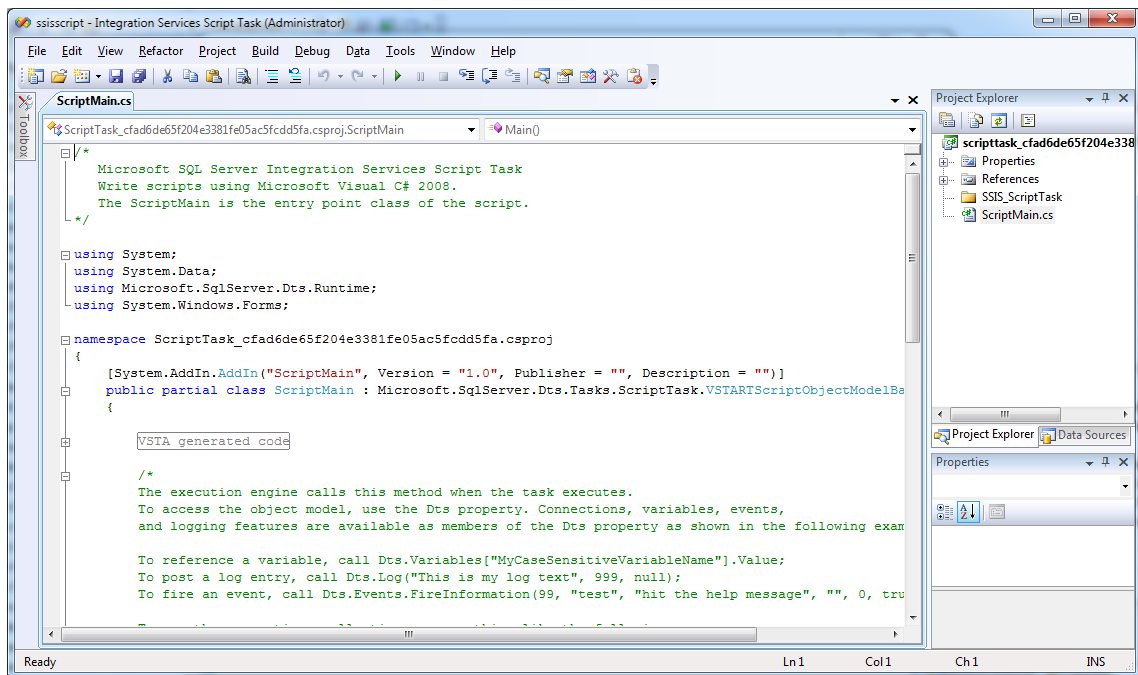


Figure 8–16. The script task allows you to write scripts using VB or C#. The text editor gives you full IntelliSense capabilities.

Remote Tasks

A remote task requires that you set up a connection manager of type HTTP. It also requires a local copy of a WSDL file to define the programmatic interface of the web service to be called. After you've met these two prerequisites, you can execute any permitted web service method as part of the control flow for your SSIS package.

Another task available in this category is the FTP task. This task allows you to configure your package to send or receive files via FTP. It also allows you to copy or move files on the remote server or local file system, or to create or delete files and folders.

SSAS Tasks

One of the most important sets of control flow tasks in the toolset for your BI project is, of course, the set of tasks that affect SSAS objects. We'll review Analysis Services Execute DDL task and the Analysis Services Processing task here. The Execute DDL task allows you to add the ability for the control flow in your package to create, alter, or drop cubes and dimensions. There are a couple of interesting considerations when configuring this new task type. The first is that two languages are used:

- *Analysis Services Scripting Language (ASSL)*: This language is used to define the particular instance of the object type that you want to affect, that is, the dimension named CustomerDim in the cube named AdventureWorks, and so on.

- *XML for Analysis (XMLA)*: This language sends the action commands, that is, create, alter, or drop, to SSAS. It does this via its execute method. The supported commands are listed in BOL under the topic “Command Element (XMLA).” Some example commands are alter, create, drop, insert, and process.

An example of using this new task type for a BI project is included with SSIS samples. Open the SyncAdvWorksPartitions solution from the samples, and then open the Sync Partitions package. The control flow for this package is shown in Figure 8–17.

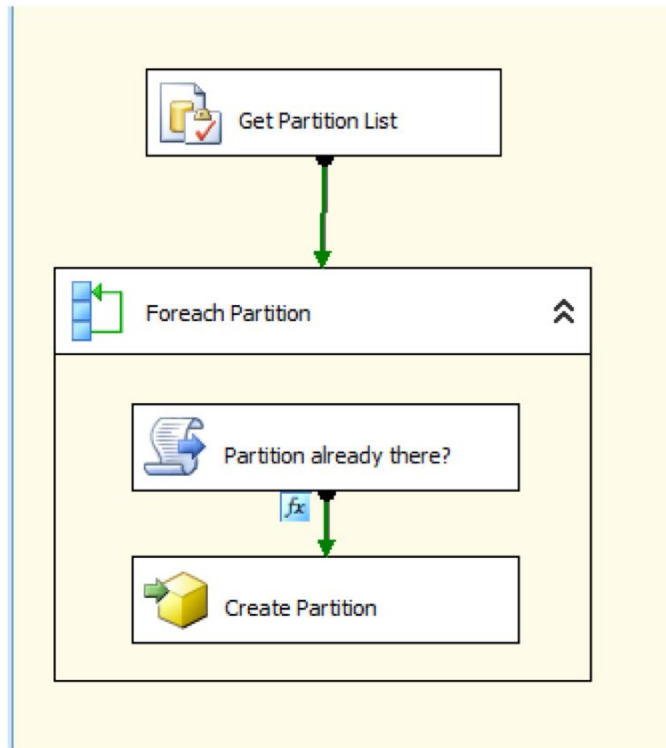


Figure 8–17. The sample package Sync Partitions shows an example of using the Analysis Services Execute DDL task to loop through the partitions and take actions appropriately.

The Execute DDL task will be one of your workhorses, especially after you move the solution into production, because you will normally want to automate the processing of cubes, dimensions, and mining models. In our experience, it is most typical to run an update-type of process on a nightly basis. This package would include success logging and failure notification. Figure 8–18 shows the configuration options for the Analysis Services Processing task.

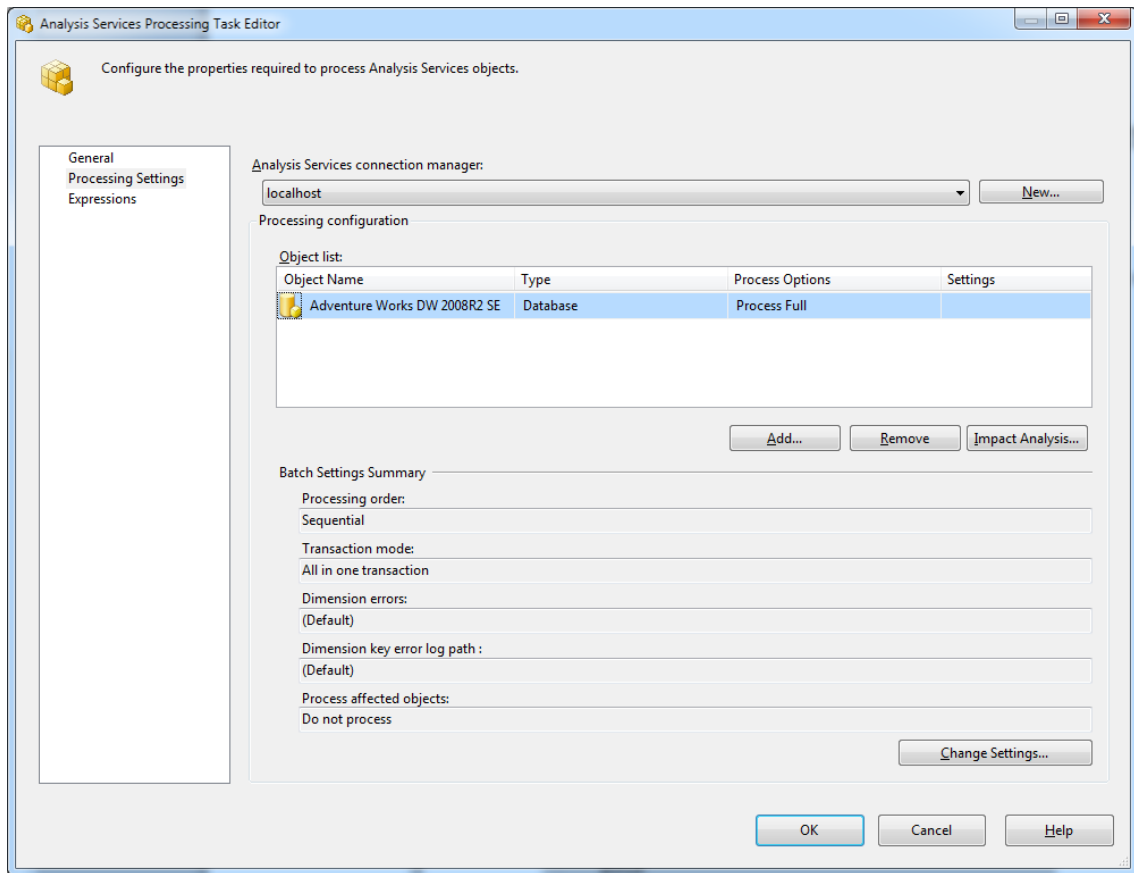


Figure 8–18. The SSAS Processing task mirrors the functionality of cube and dimension processing you have access to via the SSAS template in BIDS. In SSIS, of course, you can incorporate this task into any sort of workflow that suits your business needs.

After you've selected your particular control flow tasks, then you'll want to configure the particular workflow for these tasks. You do this by configuring the precedence constraints. These are shown on the design surface as colored lines (by default green) between control flow tasks or components.

Precedence Constraints

Precedence constraints link executables, containers, and tasks in packages into a control flow and specify conditions that determine whether control flow tasks or event handlers run. There are three types of constraints: Success (shown with a green line), Failure (shown with a red line), or Completion (shown with a blue line). To add a Success constraint between components, click the green arrow on the design surface from the source component and drag it to the destination component. Figure 8–19 shows

an example of two Control Flow components that must run successfully for the subsequent destination Control Flow component to run.

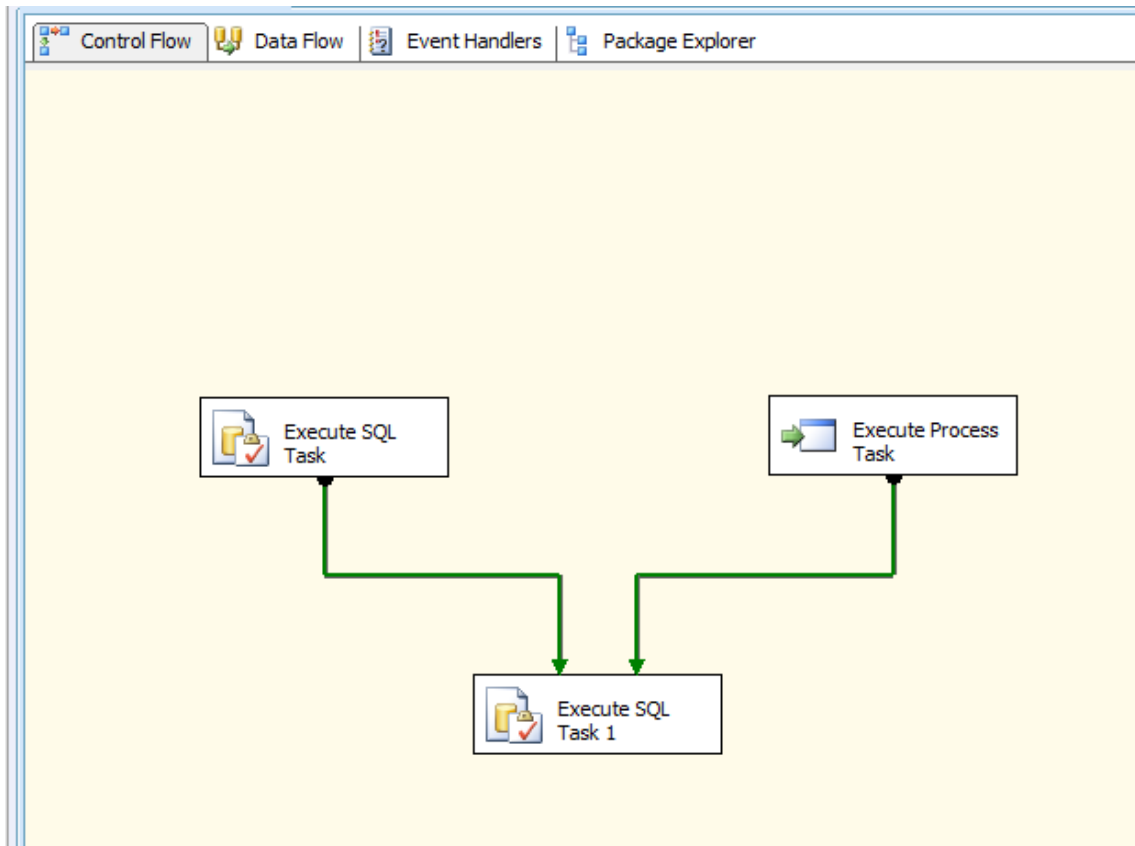


Figure 8–19. Success constraints, indicated by green lines with arrows, allow you to establish workflow paths in your SSIS packages.

The default condition when multiple constraint lines go to the same destination task is a logical AND. That means that *all* source components must execute successfully for the destination component to be allowed to execute. You can change this to a logical OR by selecting all constraint lines and then right-clicking Properties. If you then change the default setting of the LogicalAnd property from True to False, the behavior will be such that if any of the source components successfully execute, then the destination component will execute. This is shown on the design surface with dotted constraint lines in Figure 8–20.

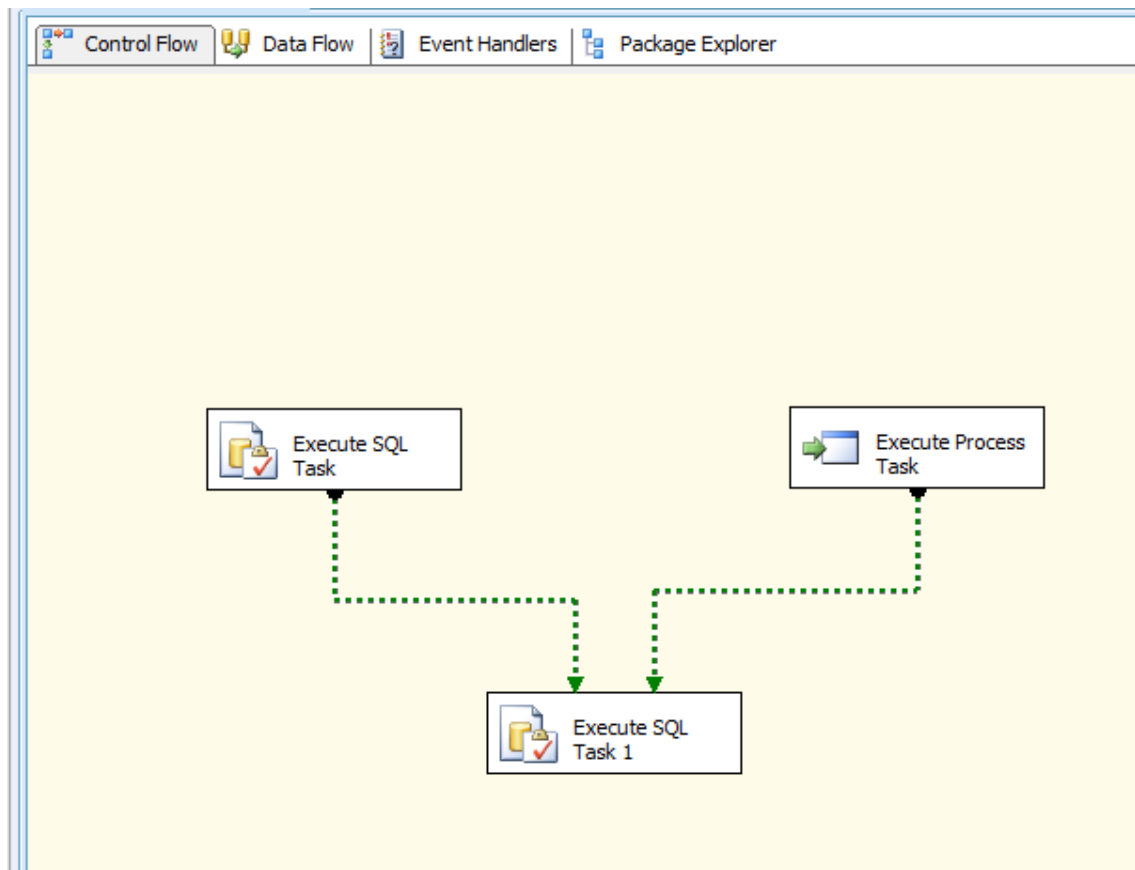


Figure 8–20. Dotted success constraint lines indicate a logical OR condition.

Using Expressions with Precedence Constraints

SSIS also allows you to associate expressions with constraints. One way to do this is to right-click the particular constraint line of interest, and click Edit. This will open the Precedence Constraint Editor dialog, which is shown with its default values in Figure 8–21.

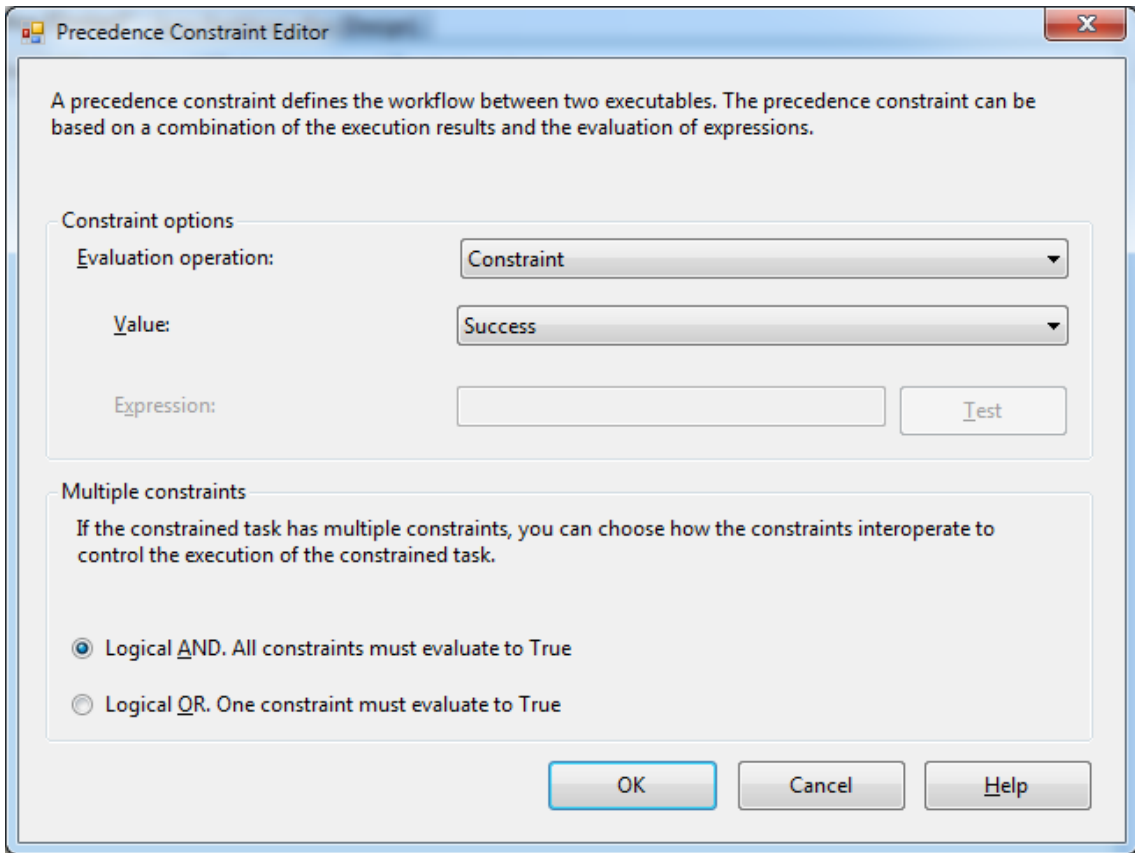


Figure 8–21. One way to alter the behavior of a precedence constraint is to right-click it and add or change the appropriate information. Note that you can add expressions to the evaluation path. You can also use the Properties window to change these values.

When combining expressions with constraints, you have these options in the “Evaluation operation” drop-down list:

- *Constraint*: This default option adds only a constraint, no expression.
- *Expression*: This adds only an expression, no constraint.
- *Expression and Constraint*: This adds both an expression and a constraint, both of which must evaluate to True to allow the destination component to execute.
- *Expression or Constraint*: This adds both an expression and a constraint, either of which must evaluate to True to allow the destination component to execute.

If you choose any of the previous three “Expression” options from the “Evaluation operation” drop-down list, the Expression text box and Test button will become available, as shown in Figure 8–22.

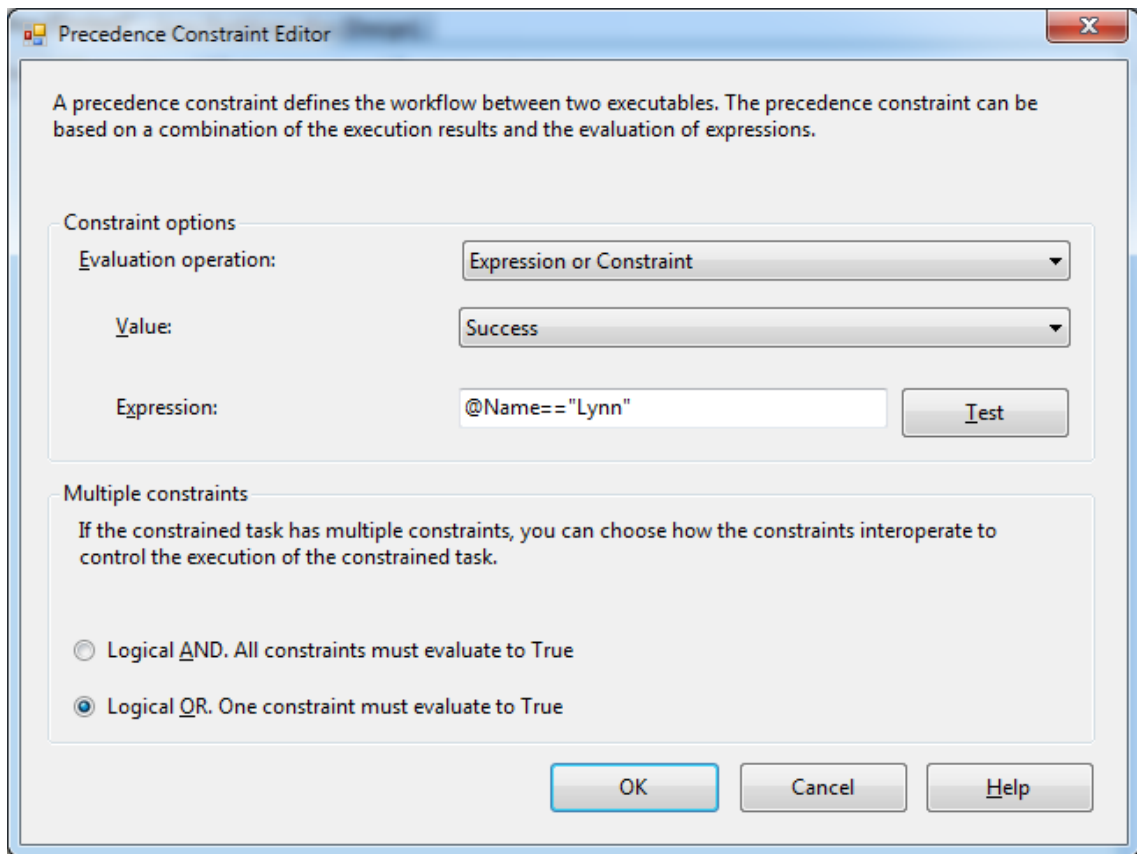


Figure 8–22. You can associate expressions with precedence constraints by configuring the Precedence Constraint Editor dialog box.

Finally, if you include an expression in a precedence constraint, BIDS will place an icon on the design surface of the control flow next to the affected precedence constraint, and the tooltip on the icon will display the configured expression, as shown in Figure 8–23. Now that we’ve reviewed control flows tasks and precedence constraints in detail, the next step is to dive in to the details of the data flow sources, destinations, and, most importantly, transformations.

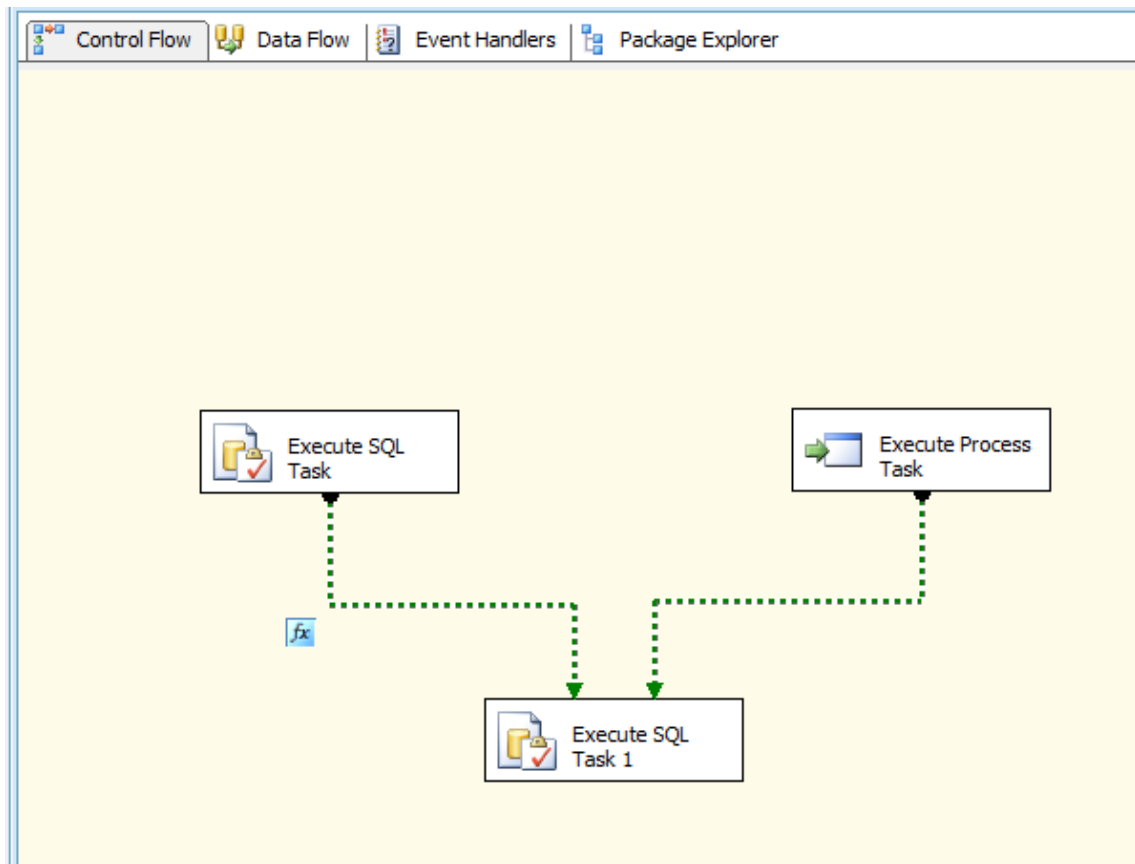


Figure 8–23. Expressions that are associated with constraints will be indicated by a small icon labeled “fx” next to the affected constraint line.

Understanding Data Flow Transformations

The Data Flow work area is where you will perform the bulk of your ETL activities. Here, you will read data sources, manipulate data by using transformations, and write your transformed data to a destination. An SSIS package can contain multiple data flows. This feature allows you to segregate your ETL tasks, for example, by business or subject area. Double-clicking a Data Flow task in the Control Flow area will take you to the Data Flow window for that particular task (shown previously in Figure 8–14).

Data Sources

To begin work in the Data Flow area, you will select at least one data flow source component. Usually, you will select at least one data flow transformation and at least one data flow destination. Figure 8–24 shows all data sources and all data destinations from the BIDS toolbox.

Some data source types to keep in mind while you build your SSIS packages are DataReader, Raw File, and XML; some destination types are DataReader, Dimension Processing, RawFile, Recordset, and SQL Mobile. The Raw File Source type is used to retrieve raw data that was previously written by the SSIS Raw File destination. It works more quickly on this type of file than using the Flat File or OLE DB Source options. You could use this source as the intermediate step in processing to more quickly move data that would be undergoing subsequent additional transformations within a package.

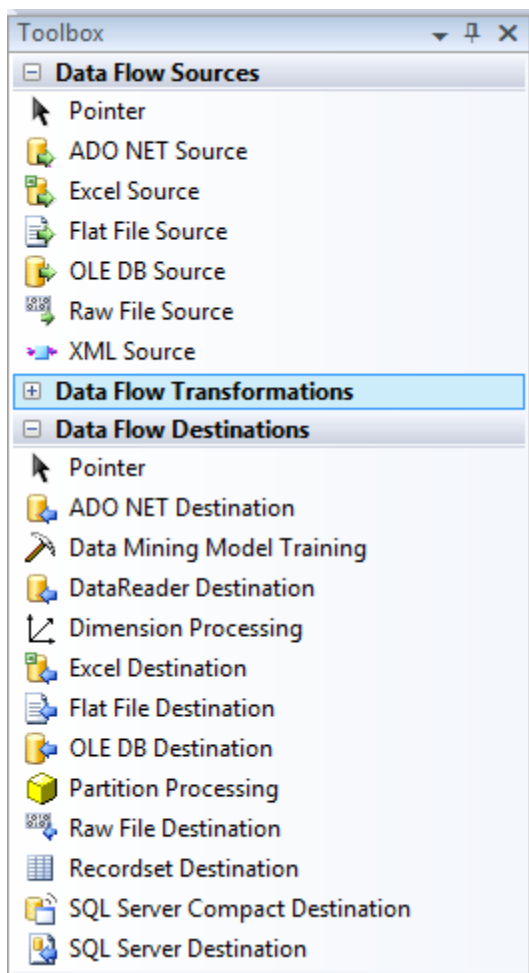


Figure 8–24. Data Flow Sources and Data Flow Destinations from the toolbox in BIDS reflect a wide variety of interoperability in SSIS packages.

The XML Source option allows you to use an XML file with or without an associated XSD schema as a source. The schema can be inline, a separate file, or generated by the task itself. XSD schemas are used to validate the format of the contents of XML files. All of these options are configured by right-clicking the task and working with the XML source editor property page, as shown in Figure 8–25.

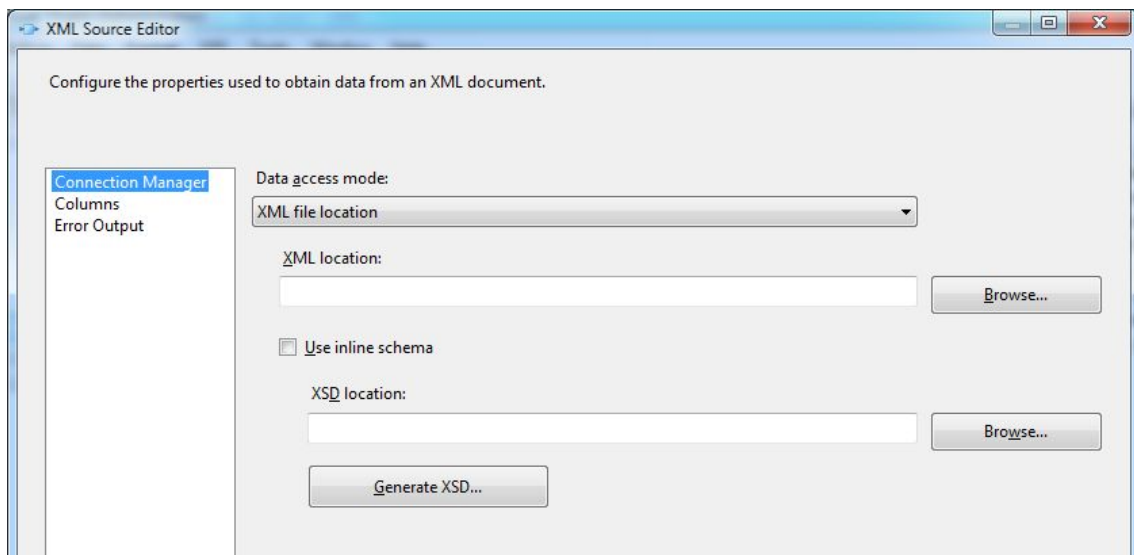


Figure 8–25. You can associate an XSD schema with an XML file as a source for your data flow by using the XML Source Editor.

Data Flow Destinations

As with Data Flow Sources, this section provides more explanation for some of the Data Flow Destination types that are used in BI scenarios. The first one to consider is the Dimension Processing destination. This allows you to map and insert the results of a data flow into an SSAS cube dimension. You'll note that you can select the Processing Method that meets your business requirements: Full, Add (Incremental), or Update. On the Mappings page, you map the source to destination columns. Finally, on the Advanced page, you set the error configuration behavior, for example, key error behavior. This last page is identical to the Advanced Processing options page available for cube and dimension processing in BIDS. The property sheet for this destination is shown in Figure 8–26.

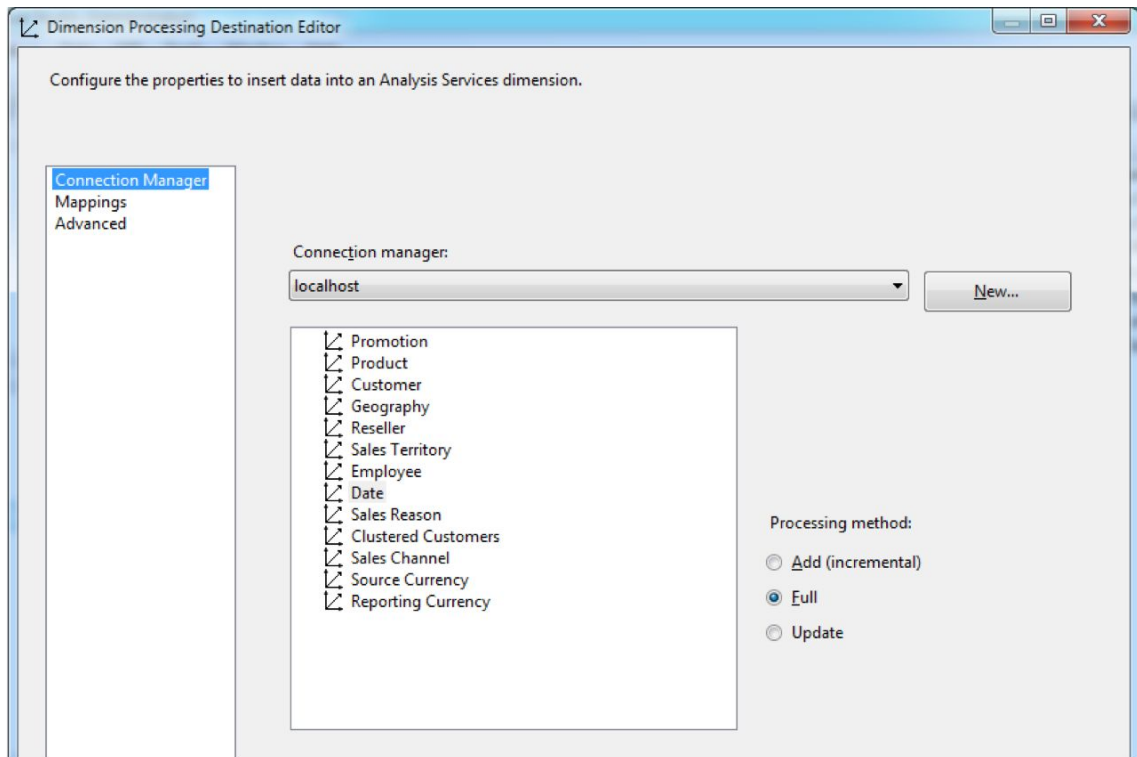


Figure 8–26. BIDS allows you to configure dimension processing, using a subset of available processing types, as a data flow destination.

The Partition Processing data flow destination presents you with a similar subset of partition processing options: Add (incremental), Full, or Data only. If your business scenario calls for a more granular type of partition processing (such as process index), you’d use the control flow Analysis Services Processing task.

Transformation Types

After you’ve added and configured your data sources and data destinations, you’ll usually then add one or more data transformations to your SSIS package. Figure 8–27 shows the transformations available in BIDS.

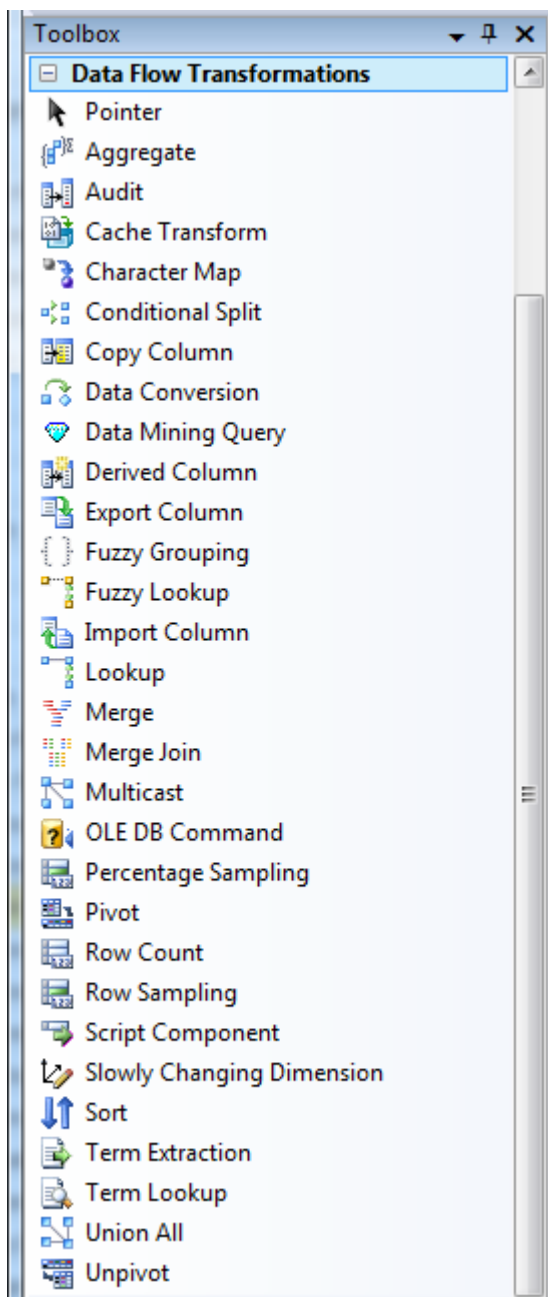


Figure 8–27. SQL Server SSIS has multiple data flow transformation types to select from.

An SSIS data flow transformation is a type of component that takes some source data, applies a type of change to any defined subset of the source data, and then makes the changed data available for output to whatever configured data flow destinations your business requirements necessitate. A great way to begin to understand the types of data flow transformations available for use in SSIS packages is to simply open the BIDS toolbox while working in the Data Flow tab for any SSIS package and pass your mouse over each transformation. When you do this, a tooltip with a brief explanation will pop up over each transformation.

If you are using the Enterprise Edition of SQL Server, you have a mind-boggling array of data transformations to use in your SSIS packages. As with control flow tasks, you can best understand what these transformations do by grouping them into categories. The following lists the transformation groups by priority order; the first group transformation types are the most commonly used in ETL projects.

- *Row*: Allows you to update data or create new data. These include Copy Column, Data Conversion, Derived Column, and Import/Export Column transformations.
- *Rowset*: This type allows you to create new rowsets. These include Aggregate, Sort, Row Sampling, Percentage Sampling, and Pivot/Unpivot transformations.
- *Split/Join*: This type allows you to perform splits, joins, and lookups on your data. These include Conditional Split, Lookup, Multicast, Merge, Merge Join, and Union All transformations.
- *Audit*: This type allows you to count rows and create auditing data. These include Audit and Row Count transformations.
- *Business Intelligence*: This type is specific to SSAS object processing. These include the Data Mining Query, and the Slowly Changing Dimension transformations.
- *Enterprise Edition*: This type is available only in the Enterprise Edition of SQL Server. These include the Fuzzy Grouping, Fuzzy Lookup, Term Extraction, and Term Lookup transformations.

As with the control flow tasks, you can extend the transformations available in the data flow transformations by downloading, purchasing, or creating your own components and then installing them in the BIDS interface.

Adding Data Transformations

The sample package `CalculatedColumns.dtsx` shows examples of the Aggregate, Derived Column, and Sort data flow transformations. The Aggregate transform allows you to easily perform aggregations of types GroupBy, Count, Count Distinct, Sum, Average, Minimum, or Maximum against your data flow by simply configuring the aggregate dialog box. The Data Flow work area for the Calculated Columns SSIS sample package is shown in Figure 8–28.

The “Add transformation” group contains some of the data flow transformation types that are most often used in BI projects: Copy Column, Import Column, and Union All. All of these transformations are self-explanatory.

The transformation group also contains the Merge and Merge Join transformations. The Merge Join transformation allows you to configure your SSIS package to perform a left, right, or full join on two sorted inputs with matching metadata. While configuring this transform, you must also specify Null handling behavior.

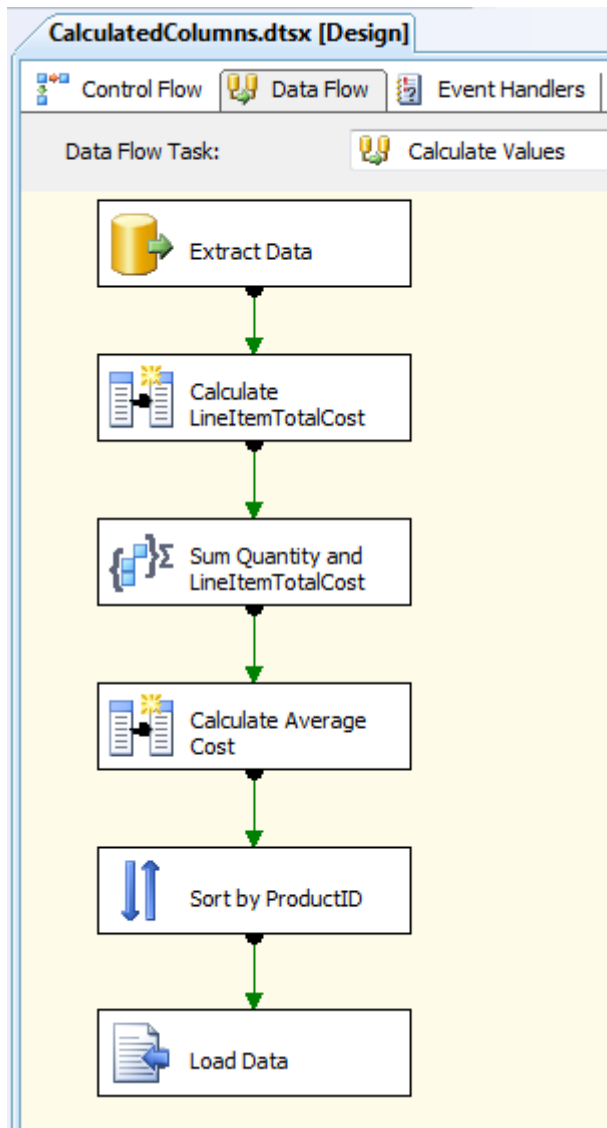


Figure 8–28. The Data Flow work area in BIDS allows you to visually configure a data flow for a particular data flow transformation in your SSIS package. This example shows the use of the Aggregate, Derived Column, and Sort data flow transformations.

Split Data Transformations

Another handy component is the Derived Column transformation. This is also shown in the CalculatedColumns.dtsx data flow sample. Figure 8–29 shows the dialog box where you configure the new column. In this work area, you can use the SSIS expression language.

Note For more information about SSIS expression syntax, see the BOL topic “Syntax (SSIS).”

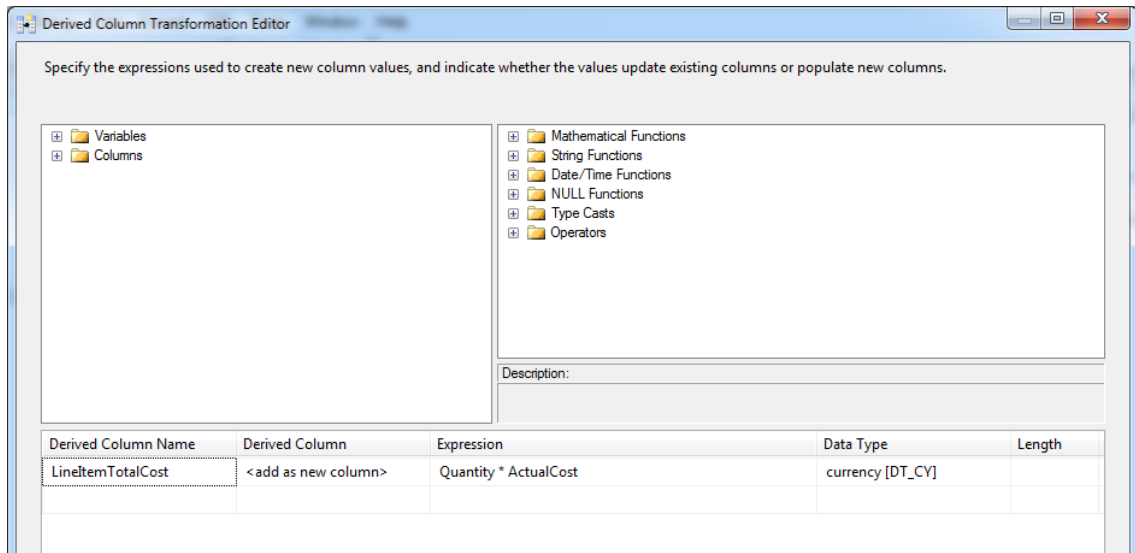


Figure 8–29. The Derived Column Transformation Editor in BIDS allows you to write expressions to create derived columns in your SSIS package.

You use the SSIS expression in your statement to derive the new column or columns. The dialog box includes a function reference for your convenience. It also includes the list of all available columns in the data flow as well as a list of all of the package variables.

The Conditional Split transformation shown in ExecutesSQLStatementsInLoop.dtsx is yet another helpful tool for you. The Conditional Split transformation routes rows to different destinations depending on the content of the data using a CASE-like decision structure. The transformation evaluates expressions and, based on the results, directs particular data rows to the specified outputs. The data flow for this package is shown in Figure 8–30, and the Conditional Split dialog box is shown in Figure 8–31.

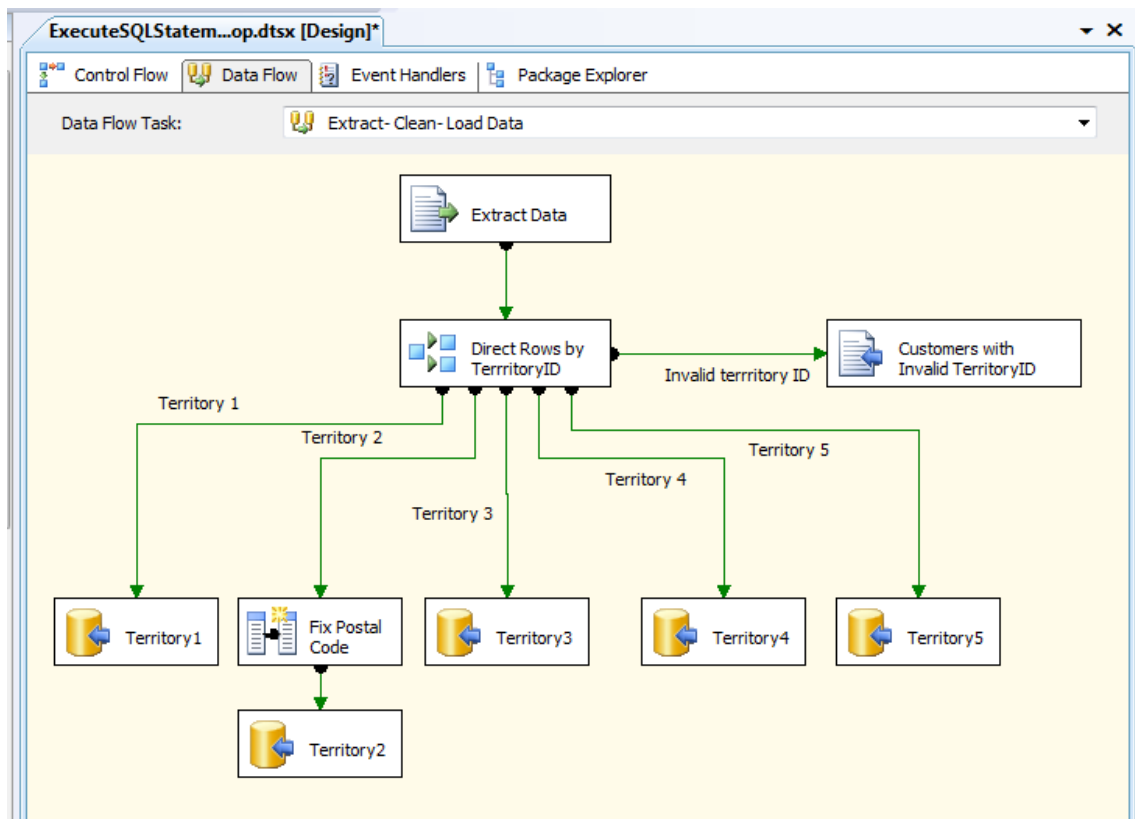


Figure 8–30. The sample package *ExecuteSQLStatementInLoop.dtsx* shows off the Conditional Split data flow transformation type.

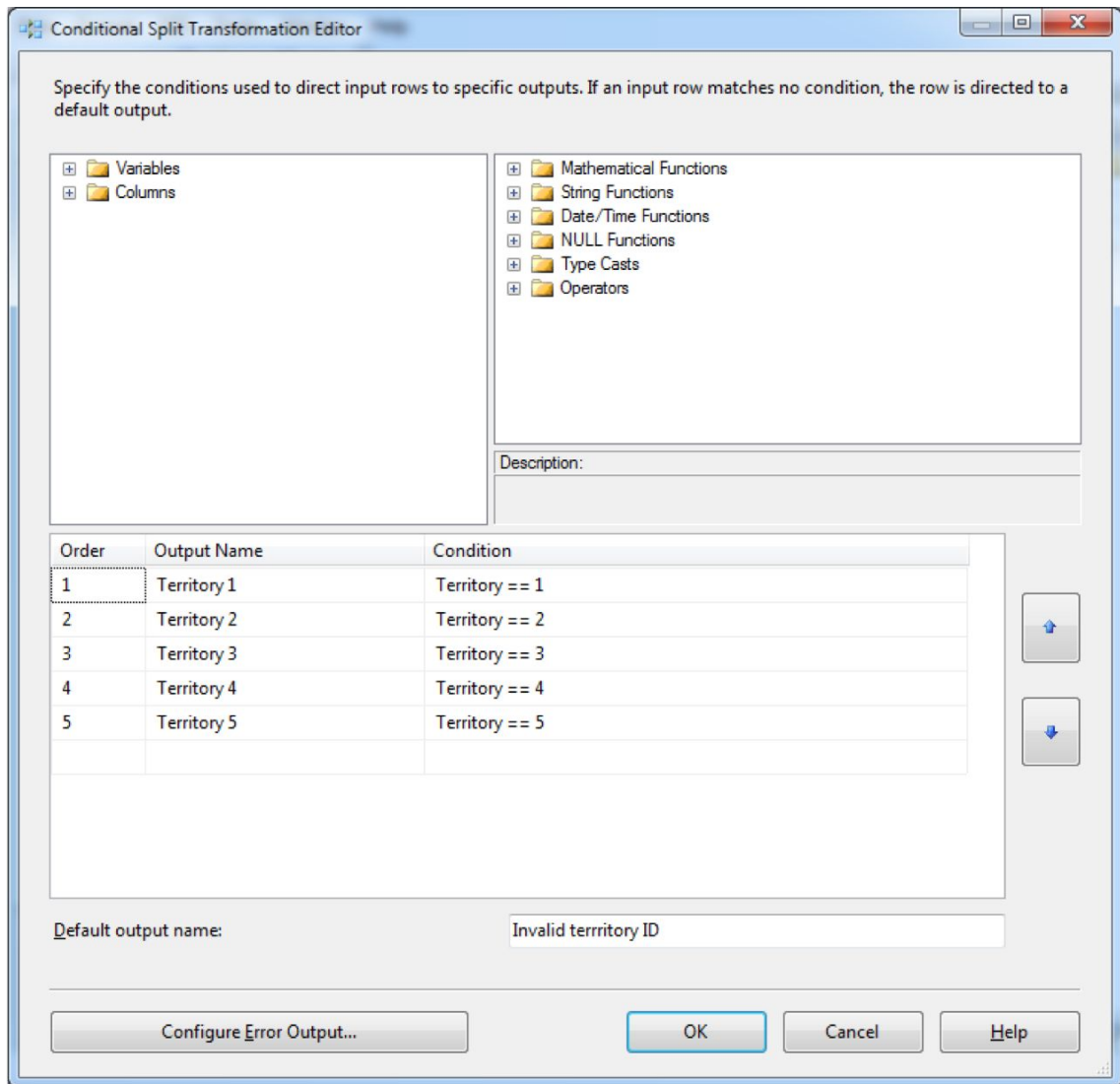


Figure 8–31. The Conditional Split Transformation Editor dialog box allows you to specify the split conditions via expressions.

Translate Data Transformations

The Lookup and Union transformations are showcased in the complex sample `DataCleaning.dtsx`. The Lookup transformation performs lookups by joining data in source columns with columns in a reference table.

The Fuzzy Lookup transformation is contrasted to the regular Lookup transformation because it uses fuzzy matching. The Lookup transformation returns either an exact match or nothing from the reference table. The Fuzzy Lookup transformation uses fuzzy matching to return one or more close matches from the reference table.

■ **Tip** A Fuzzy Lookup transformation frequently follows a Lookup transformation in a package data flow (as is shown in the sample package). First, the Lookup transformation tries to find an exact match. If it fails, the Fuzzy Lookup transformation provides close matches from the reference table.

The Data Cleansing sample is the most complex of the samples. It includes not only the two transformations mentioned previously but also uses many others, including the very powerful Fuzzy Lookup and Fuzzy Grouping. The fuzzy transformation types require the Enterprise Edition of SQL Server. This sample is also a very good example of using annotations on the design surface to make SSIS packages self-documenting. Figure 8–32 shows the data flow for this package.

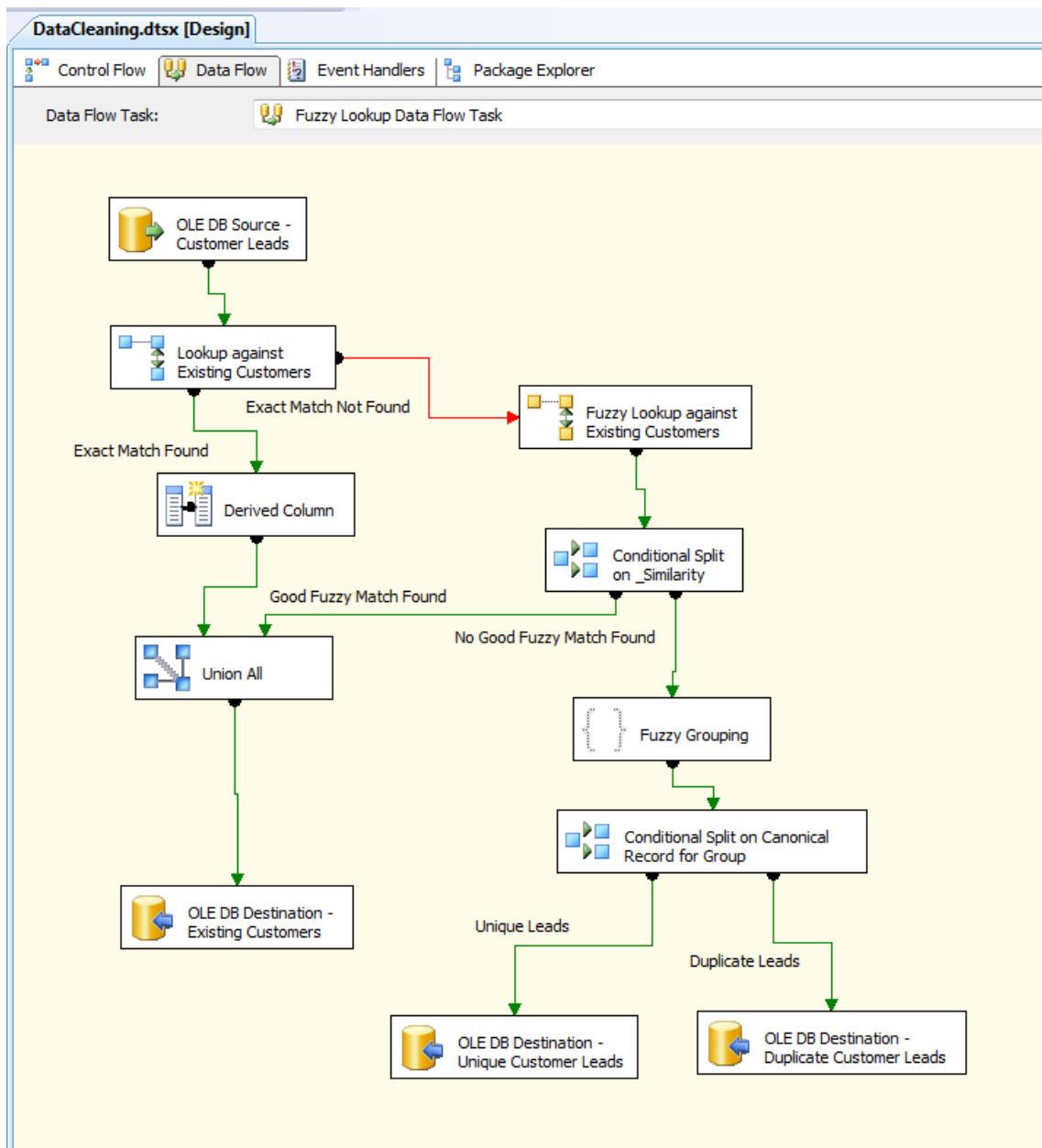


Figure 8–32. The DataCleansing sample SSIS package uses the most complex data flow sequence of the samples included with SQL Server. It includes the following data transformation types: Lookup, Derived

Column, Union, Conditional Split, Fuzzy Lookup, and Fuzzy Grouping. It is also a good example of documenting functionality on the design surface via annotations.

The Lookup transformation has some configurable capabilities via the Advanced property sheet. The new functionality allows you to more granularly control the overhead associated with the performance of the lookup activity. You configure this on the Advanced tab of the Lookup Transformation Editor dialog box. The Advanced tab is shown in Figure 8–33.

Tip Lookup transformations are case-sensitive. You can use UPPER or LOWER T-SQL functions to convert lookup source data to the correct case prior to using them in a Lookup transformation.

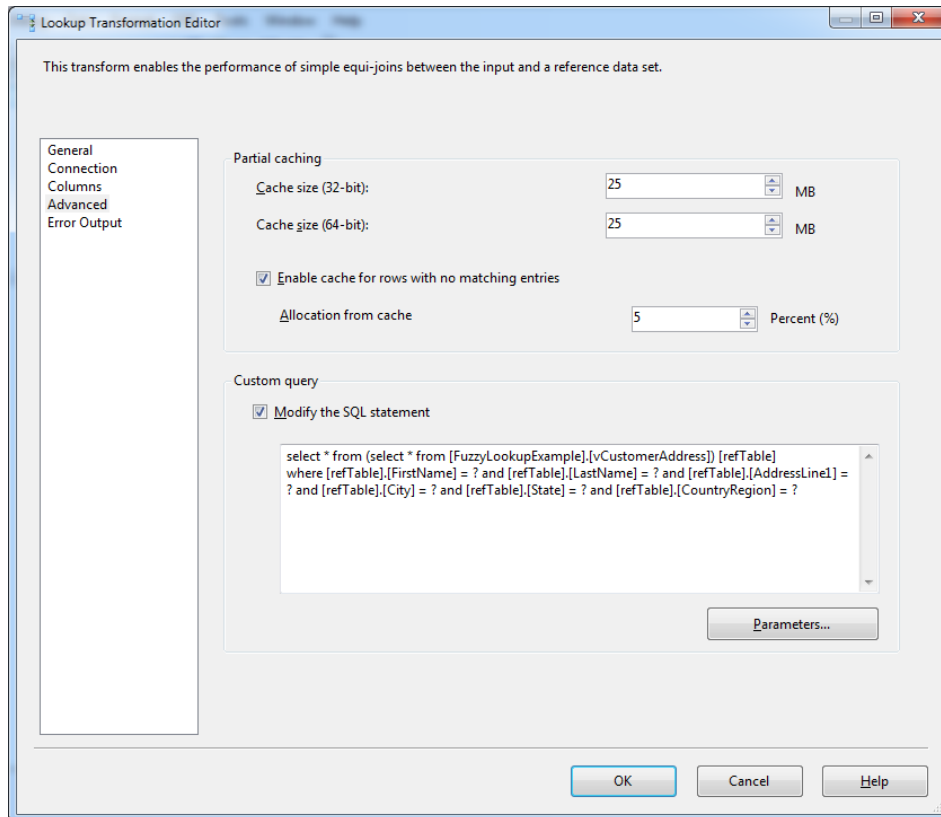


Figure 8–33. The advanced tab of the Lookup Transformation Editor allows you to configure additional settings on it. This allows you greater control over the performance overhead generated by this transformation.

SSAS Data Transformations

An interesting SSIS sample package showcases the Audit transformation from the SSAS transformation category. This package also demonstrates a common BI scenario: tracking data lineage. Lineage is a formal way for saying that your business requirements include the need to track package execution history. The `CaptureDataLineage.dtsx` package includes an Audit transformation. Part of the Audit Transformation Editor dialog box is shown in Figure 8–34.

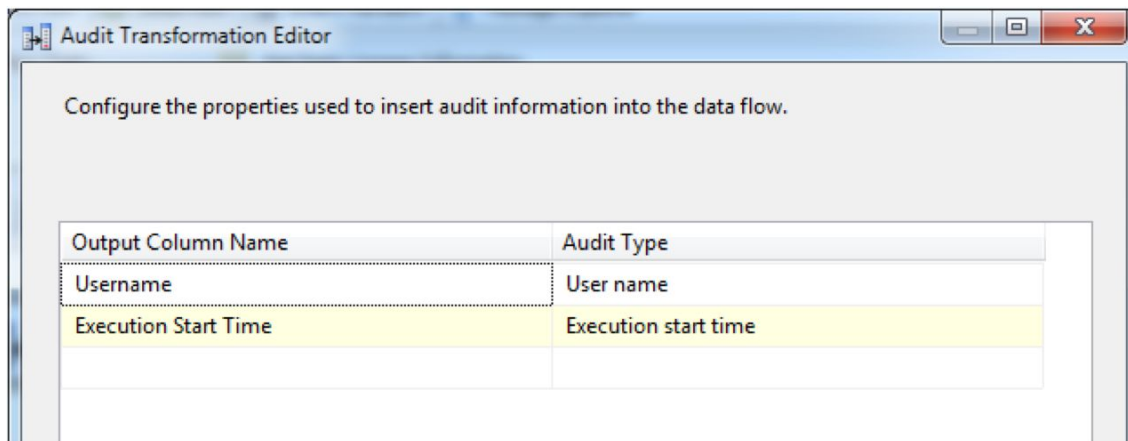


Figure 8–34. The Audit data flow transformation allows you to capture lineage or package execution information easily for logging.

Slowly Changing Dimension Transformation

The Slowly Changing Dimension (SCD) transformation is probably the most important data transformation in the toolbox for BI solutions. Using it properly requires that you understand the concepts behind changing dimensions (type 1, 2, or 3) and what structures are required to support them. When using this transformation type, you'll have to configure several options via the Slowly Changing Dimension Wizard. To open the wizard, you select a Slowly Changing Dimension transformation, drag it onto the data flow SSIS design surface, right-click the transformation, and then click Edit. The first choice for you to make in the wizard is to select the Business key from the Key Type column of the source table as shown in Figure 8–35.

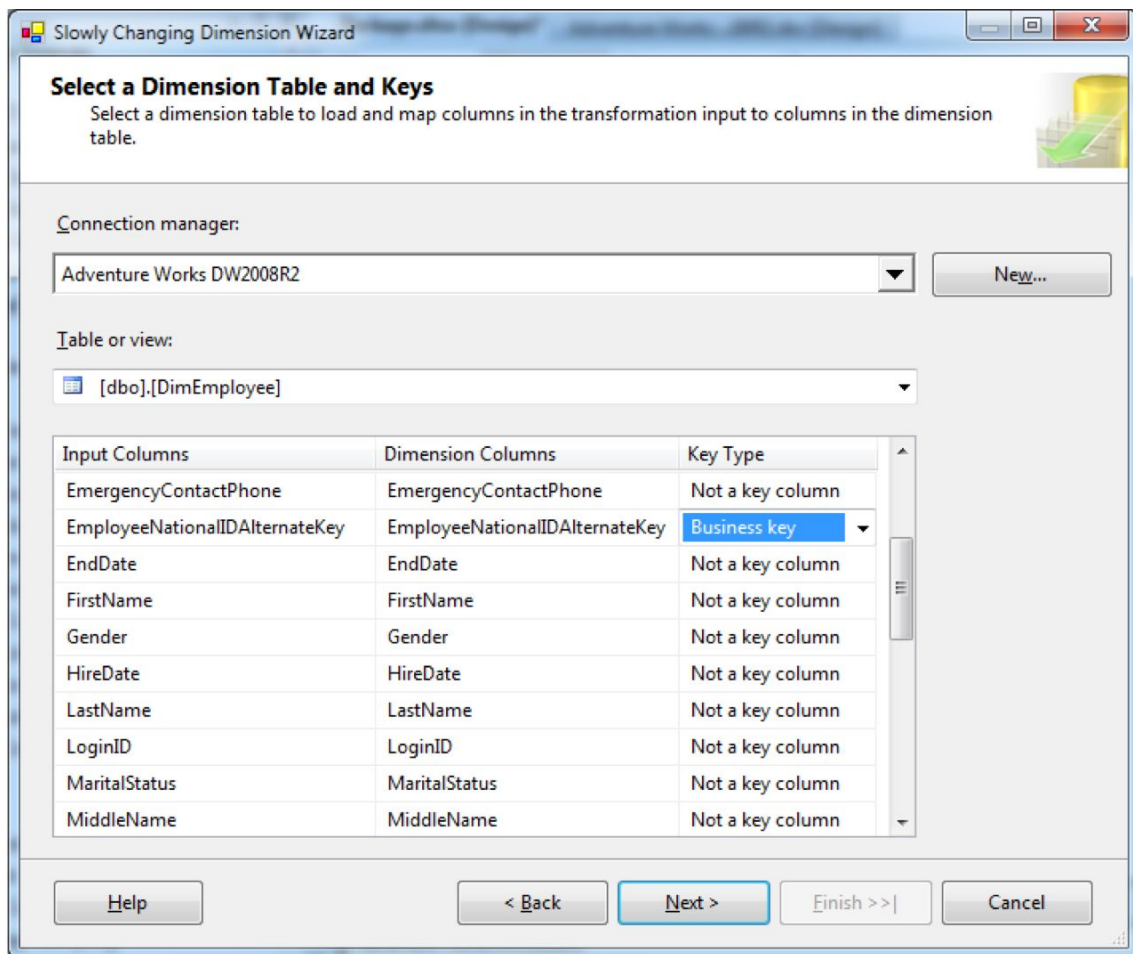


Figure 8–35. When using the Slowly Changing Dimension data flow transformations, the first step is to select the source table and its Business key.

The next step is to configure the change type for each attribute in the table. The choices are Fixed Attribute (Do not overwrite, treat as error), Changing Attribute (Overwrite, no history), or Historical Attribute (write new record and mark previous values as outdated).

This corresponds to SCD Type 1 for maintaining the records current state and SCD Type 2 for maintaining historical records. SCD Type 3, which adds an additional dimension attribute for each change, is *not* supported by this transformation type. These options are shown in Figure 8–36.

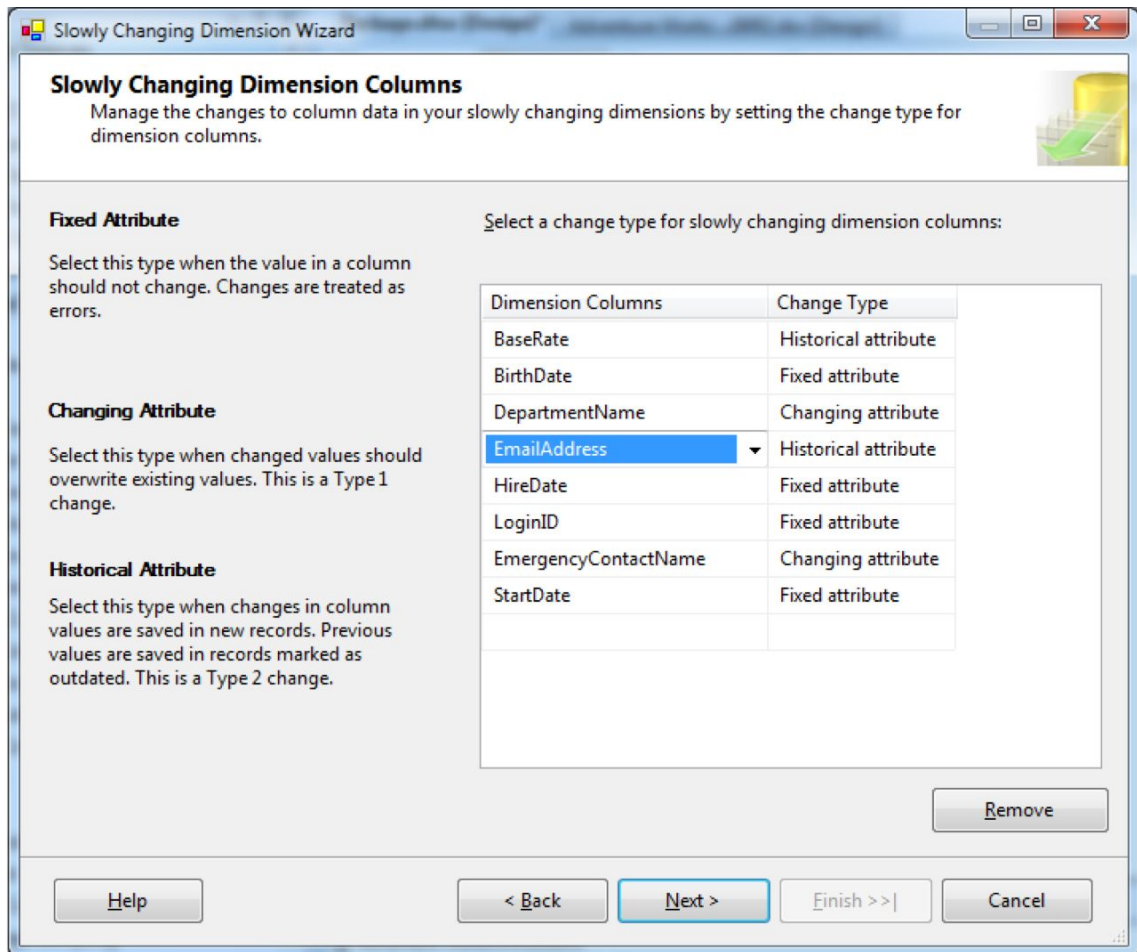


Figure 8–36. When using the SCD data flow transformation, the second step is to configure the change type for each attribute of the dimension table.

The next step is to configure the optional behavior for fixed attributes and changing attributes. These options, which include failing the transformation if changes are detected in a fixed attribute and/or cascading changes to all affected records for attributes marked as changing type, are shown in Figure 8–37.

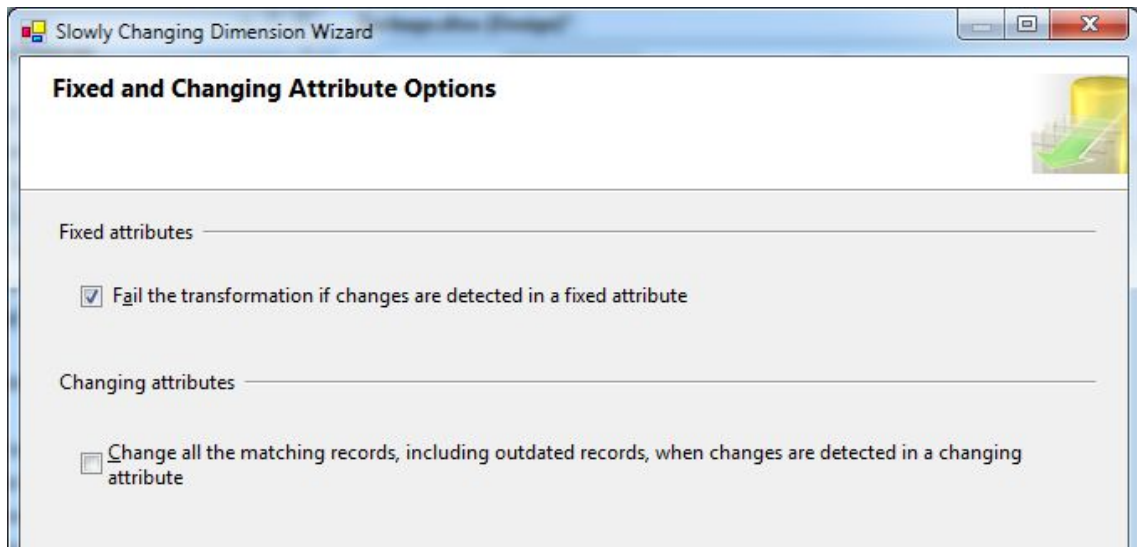


Figure 8–37. When using the SCD data flow transformation, the third step is to configure the optional behavior for fixed and changing attributes.

The next step is to configure the storage location for historical attribute values. These options are shown in Figure 8–38.

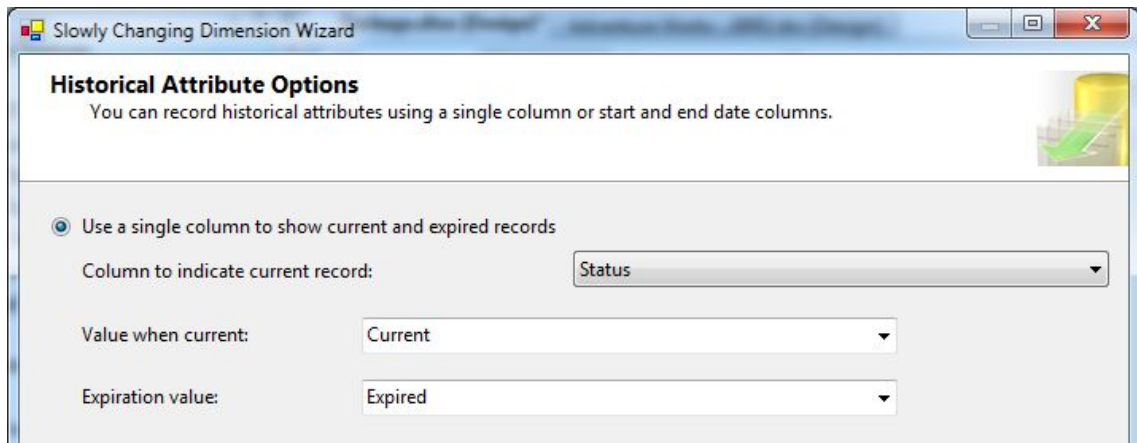


Figure 8–38. When using the SCD data flow transformation, the fourth step is to configure the storage location and values for historical attributes.

The next step is to configure the behavior for inferred dimension members. These options are shown in Figure 8–39. The SCD wizard is an incredibly powerful and useful tool. It has been designed to help you quickly implement the most common SCD scenarios from your particular BI solution’s requirements.

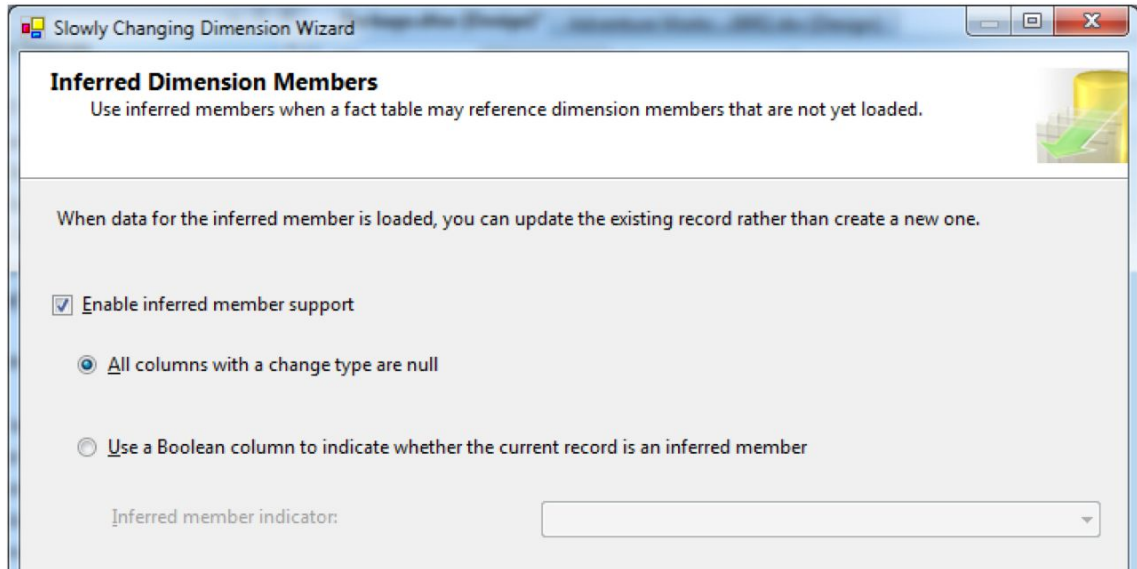


Figure 8–39. When using the SCD data flow transformation, the fifth step is to configure the behavior for inferred members via the options in the wizard.

Sample Data Transformations

The Percentage Sampling, Row Count, and Row Sampling transformations included in this group are pretty straightforward to use. They all allow you to sample or to count and to send output, if sampling, to a configured destination. You configure the number (or percentage) of rows to be sampled, and you can also configure a random seed value via the property sheet.

These transformations can be particularly useful in BI projects because you may be working with huge volumes of data. Using transformations from the Sample data group can help you to quickly (that is, with very little overhead) get a view of what you are working with.

Run Command Data Transformations

The new script component transformation requires a bit of explanation. Why the need for an additional script component transformation in the data flow transformations area? You’ll see that the key to understanding the difference between the Script task and the script component is to be found by looking at the first dialog box that you must configure when using the script component. In this dialog box shown in Figure 8–40, you must select whether the component will be used to associate a script with a data flow source, destination, or transformation.

Because the Script Component output becomes part of the data flow, rather than the more general output of the more general control flow Script task, there are several syntax differences when writing your scripts between these two transformations. For a more detailed discussion, see the BOL topic “Comparing the Script Task and the Script Component.”

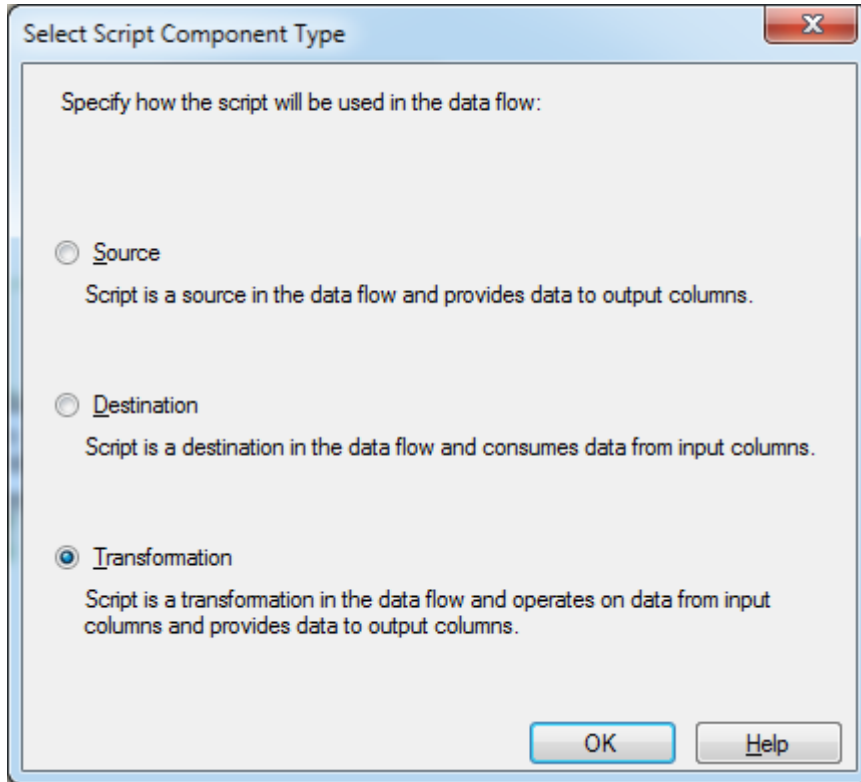


Figure 8–40. The Script Component data flow transformation requires that you associate the VB.NET script with a data flow source, destination, or transformation.

Enterprise Edition–Only Data Transformations

The Fuzzy Logic, Fuzzy Grouping, Term Extraction, and Term Lookup transformations are only available in the Enterprise Edition of SQL Server. Both types of fuzzy transformations are shown in the sample package named `DataCleaning.dtsx`. The data flow for this package was shown earlier, in Figure 8–32.

■ **Note** What does “fuzzy” mean? In a word, “inexact.” The difference between a typical database lookup and a fuzzy lookup is that the database lookup is Boolean; that is, there is a match to a string, to a portion of a string, or no match at all. Fuzzy lookups and groupings use a much more sophisticated algorithm that finds a candidate match. The result is that lookups and groupings will be more flexible, inclusive, and ultimately useful.

Term Extraction tasks and Lookup tasks use the same powerful fuzzy type of logic in your package’s data flow. The Advanced property sheet for the Term Extract transform type allows you to even more finely configure the type of term you are looking for, that is, whether it’s a noun or not. Your input will impact the algorithm because different linguistic pattern matching is used for different parts of speech. For example, nouns are searched for both in singular and plural format; verbs are searched for using all possible declensions. Figure 8–41 shows the Advanced property sheet for the Term Extraction transformation.

A primary consideration when using the wide variety of transformations now available in BIDS is the execution overhead and complexity that you are adding to your package. Execution overhead includes stressing processors, memory, and IO. The components most affected depend on the type of transformations, quantity of data, and other factors. You can use the Windows System Monitor Counter Object `SQLServer:SSIS Pipeline` and its associated 12 counters to help to determine whether and where you may have bottlenecks.

As with SSAS advanced cube design options, advanced transforms, such as Fuzzy Lookup, should only be used when business needs call for them, and they should always be tested with production levels of data to ensure that production systems can process the required loads given the resources that they have available.

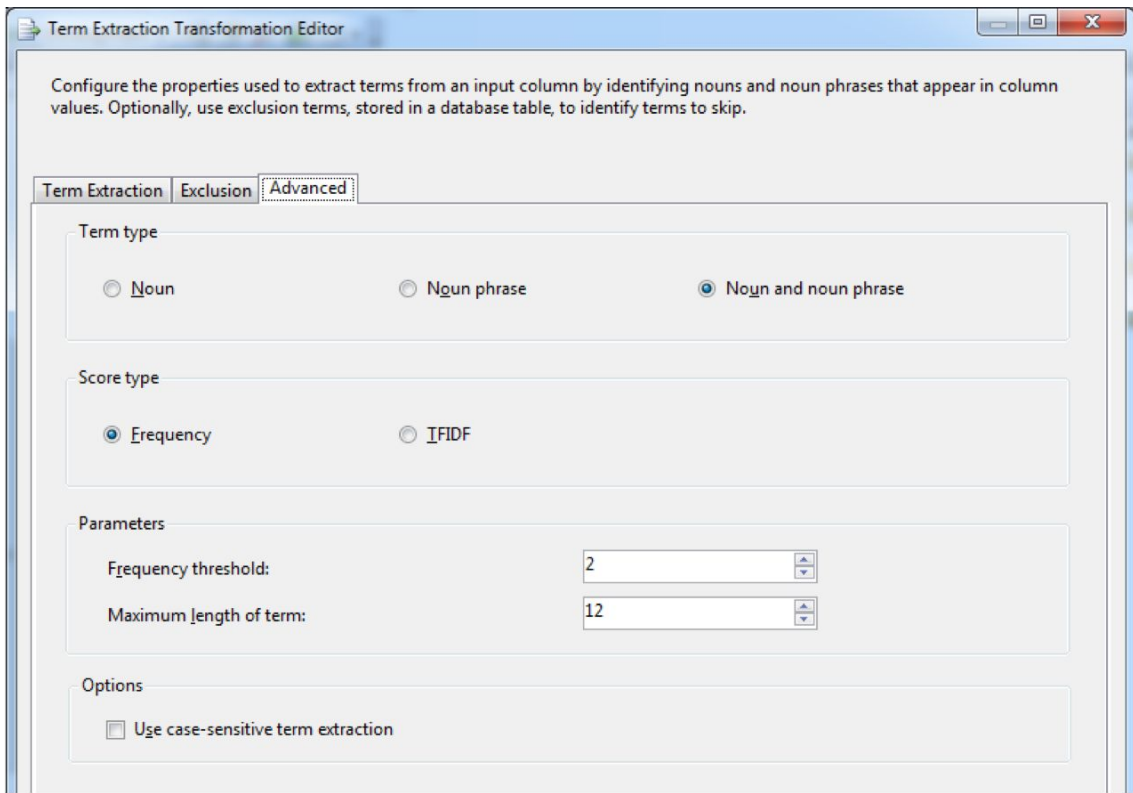


Figure 8–41. The Term Extraction Transformation Editor allows you to more granularly configure this transformation. In the Score type section, TFIDF refers to “Term Frequency and Inverse Document Frequency”.

Using the Dynamic Package Configuration Wizard

An important, and practical, consideration when implementing SSIS packages is creating packages that can easily be moved from a development environment to a production environment. As stated in the “General ETL Package-Design Best Practices” section of this chapter, this means that many package values, such as connection string, should never be hard-coded into a package. The Package Configuration Wizard will help you accomplish this goal.

To access this wizard, you choose SSIS ► Package Configurations from the BIDS menu bar. Once in the wizard, you can set most properties of an SSIS package via a number of dynamic methods. Figure 8–42 shows the wizard’s interface for selecting the property you want to configure dynamically.

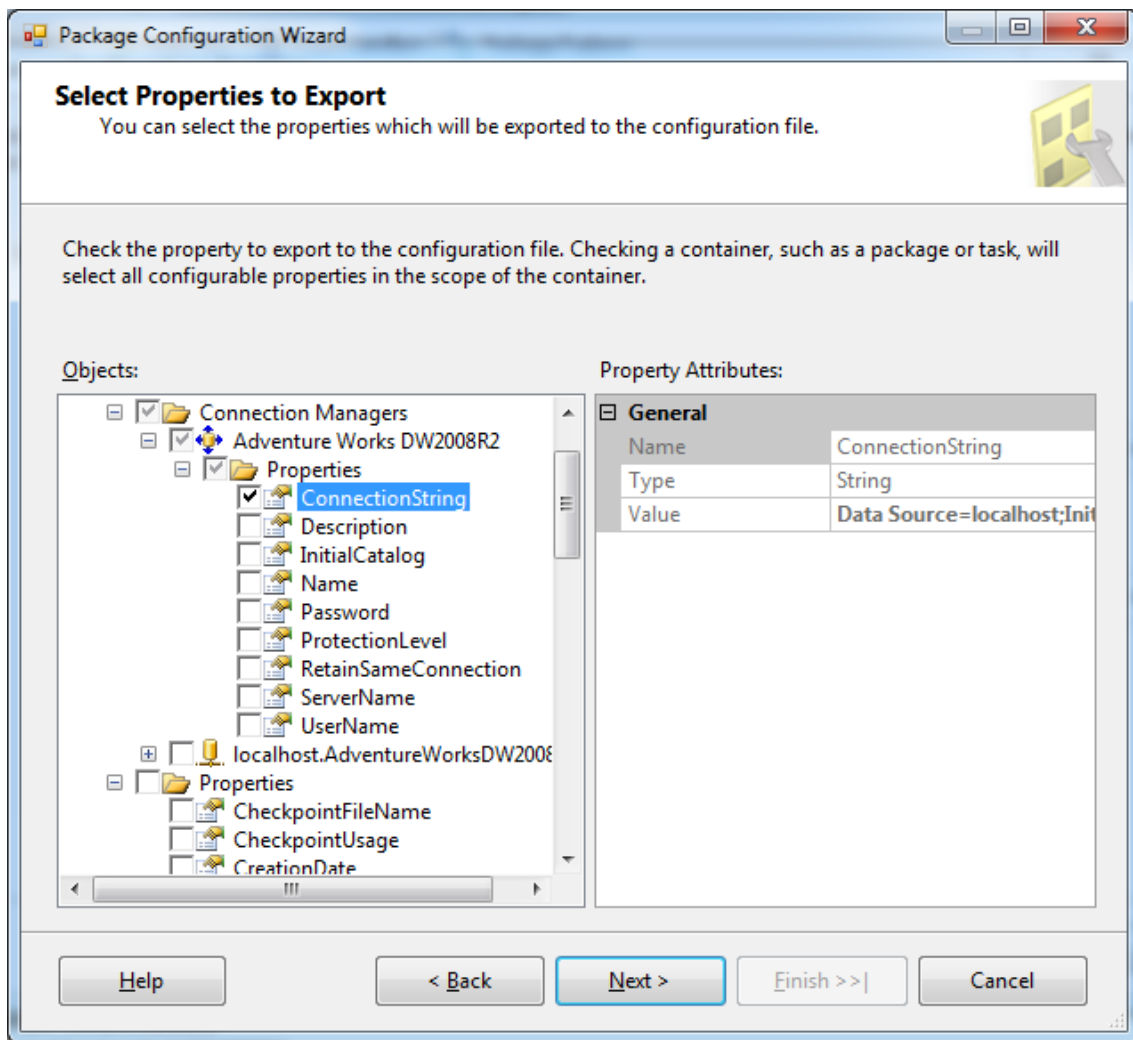


Figure 8–42. The Package Configuration Wizard allows you to set SSIS package properties dynamically by showing you a hierarchical view of all objects (such as Connection Managers, Properties, Variables, and Executables) contained in the package.

The wizard provides you with a number of options to set whatever properties you are working with dynamically. An example of a property that you may choose to use this functionality to configure could be the connection strings values for data sources or destinations. The sources for property values can originate from any of these options: XML configuration file, Environmental Variable, Registry Entry, Parent Package Variable, or SQL Server.

Assigning SSIS Expressions

In some ways, I've saved the best for last; SSIS has the ability to configure expressions for nearly every control flow task type property value. You learned earlier in this chapter that SSIS expressions could be used in conjunction with control flow precedence constraints; they can also be used with particular control flow tasks. So why are expressions associated with tasks or components so wonderful? These expressions can be assigned to most any read/write property in the entire SSIS package, which, in effect, enables dynamic update of the associated property at runtime. Figure 8–43 shows the interface for expression definition using the Execute SQL control flow task.

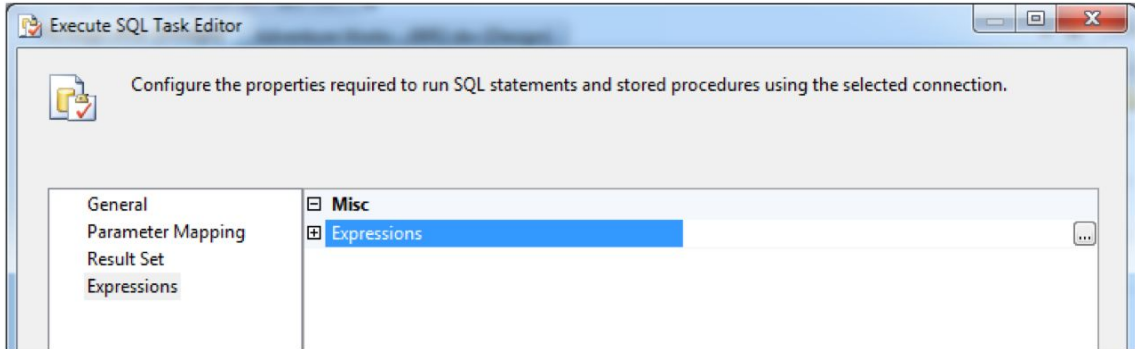


Figure 8–43. The Expressions section of control flow tasks allows you to dynamically set read/write property values.

One common use of property expressions is to allow for dynamic update of a connection string using a value that is stored in a variable. Expressions can be added to a package, task, event handler, connection manager, log provider, or (as discussed earlier in this chapter) precedence constraint. Event handlers and log providers are covered in Chapter 9, “Advanced SSIS.”

Summary

SSIS provides you with an incredible degree of flexibility and power when implementing your ETL solution. This chapter covered the foundations of using the tools available in BIDS to create the SSIS packages that will be the workhorses of your ETL processes for BI. You learned more about common objects—data sources and DSVs—and then explored connection managers. The heart of this chapter covered the core of the ETL toolset—control flow task and data flow transformations. You also learned about using the Dynamic Package Configuration Wizard.

Believe it or not, there is still much more to learn about creating data workflows using SSIS. In Chapter 9, we'll explore debugging, error handling, event handling, logging, and using transactions in SSIS packages in BI projects.



Advanced SSIS

We just completed an extensive tour of SSIS, including most all of the design surfaces in BIDS. What else could we possibly have to cover in the SSIS area? Actually there is still a great deal more to learn. The focus of this chapter is on using BIDS to make your packages even more useful by adding or using features that will improve your ability to reuse, deploy, and debug them. Also, we'll look at error handling and some new SSIS features.

In this chapter, we'll cover the following topics:

- Understanding package execution—viewers and BIDS
- Debugging—breakpoints, watch windows, and more
- Logging—new log providers and error handling
- Deploying packages
- Setting runtime properties
- Defining security
- Placing checkpoints
- Using transactions
- Data profiling

Understanding Package Execution

The BIDS environment is not only conducive to implementing elegant data workflow packages but it also has a large number of features that allow you to understand the implications of package execution.

The most basic of these tools are the Package Explorer view, the package Progress window view, and the design surface itself. When you execute packages in BIDS, all of these areas dynamically reflect a large amount of important information about each step in the package as it executes. The Package Explorer view shown in Figure 9-1 lists all the objects associated with the DataCleaning sample package.

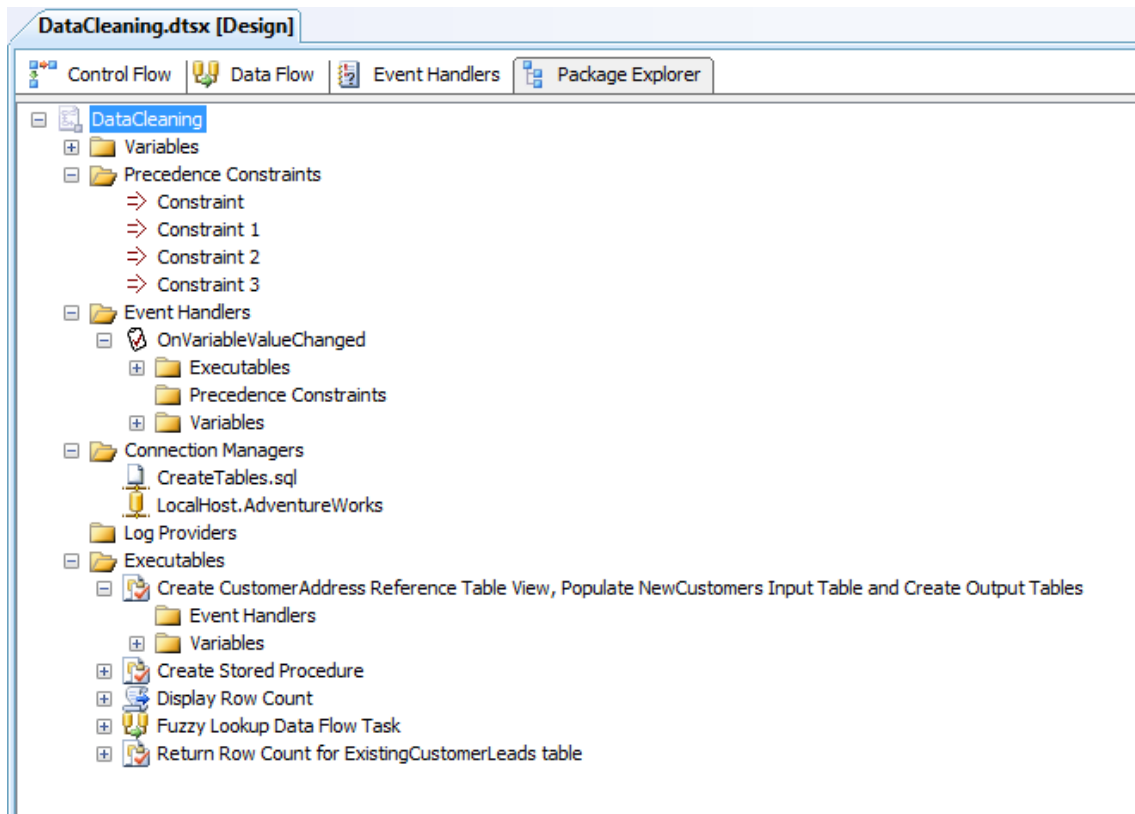


Figure 9–1. The Package Explorer view shows a complete list of objects in a particular package. Objects are categorized and grouped into folders by type of object.

The next areas for you to work with to get more insight into the execution overhead associated with your package are the Control Flow work area and Data Flow work area. As you execute your package by clicking the green triangle on the toolbar (or by right-clicking the package file in Solution Explorer and then clicking Execute Package), each task item on the Control Flow and transformation on the Data Flow work areas changes color to indicate status: yellow for executing, red for failure, and green for success. Also the Data Flow area shows the number of rows output by each transformation (see Figure 9–2).

Note You cannot make any changes to the package while it is executing. To alter the package, you must stop the execution by clicking the Stop Debugging button (the small, square, blue button on the top toolbar).

Another informational indicator available to you in BIDS is the Progress window. When you start a package executing, the Progress tab will become available at the top of the SSIS design area. This window shows you detailed information about the runtime details associated with executing each step of your package. Events, warnings, and errors are included in this information. A typical Progress window is shown in Figure 9–3.

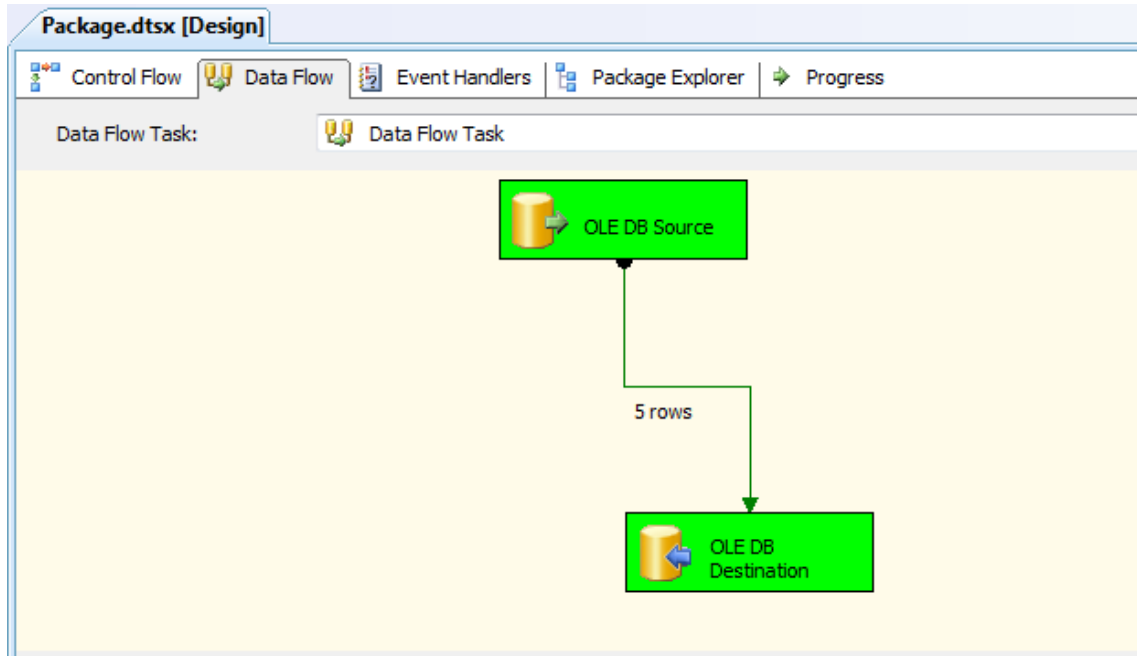


Figure 9–2. As you execute SSIS packages in BIDS, you can view the progress of each task on the Control Flow design surface and each transformation on the Data Flow design surface. BIDS will also report the number of rows output for each transformation on the design surface.

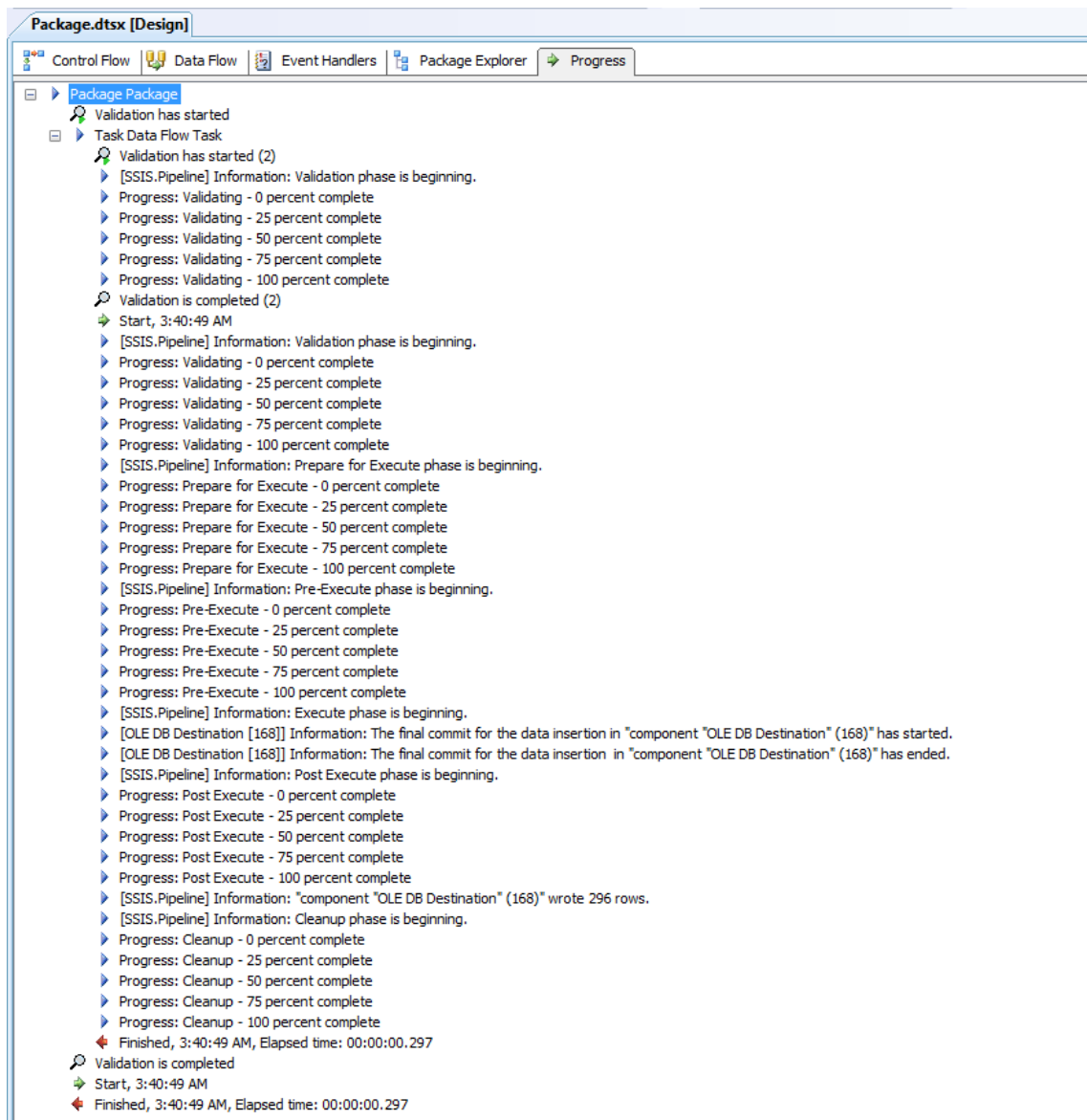


Figure 9–3. The Progress window reports detailed information about the execution of each task. It also reports events, warnings and errors.

Data Viewers

A *data viewer* is a visual tool that will help you understand your data flow. Data viewers can be added to one or more paths in your package's data flow. To add a data viewer, you right-click a data flow path arrow (which is represented as a green line) that connects any data flow transformations. You then click Data Viewers. You can insert multiple data viewers into any one section of your data flow. Each data viewer can have its own configuration settings.

In the configuration settings, you select one or more columns from the data flow to include based on the type of viewer you've selected. Figure 9–4 shows a configuration that uses four possible data viewer types: Grid, Histogram, Scatter Plot, and Column Chart.

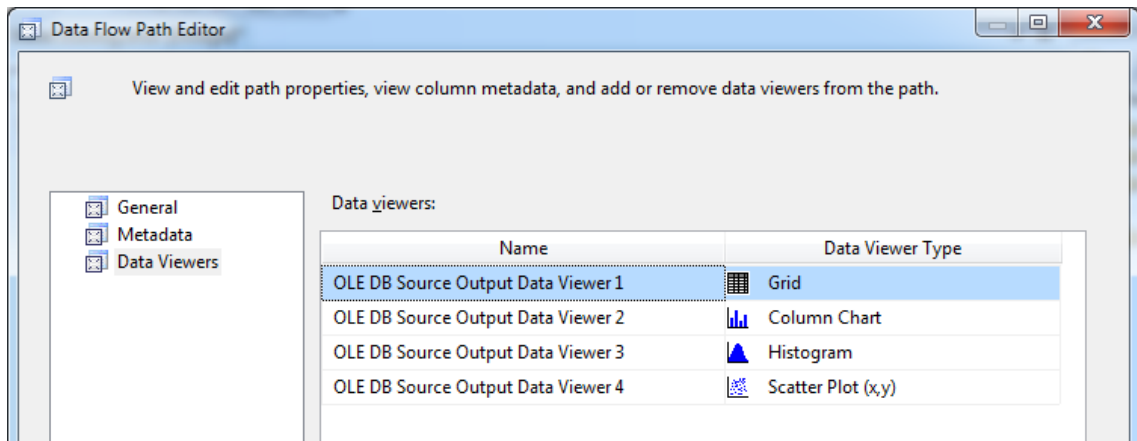


Figure 9–4. The Data Flow Path Editor allows you to add multiple data viewers to a data flow. You can add Grid, Histogram, Scatter Plot, or Column Chart data viewers.

After you've added a data viewer to a particular data flow, BIDS indicates this on the Data Flow design surface by placing a small table-shaped icon (with glasses) next to the data flow path arrow where the data flow has been configured, as shown in Figure 9–5.

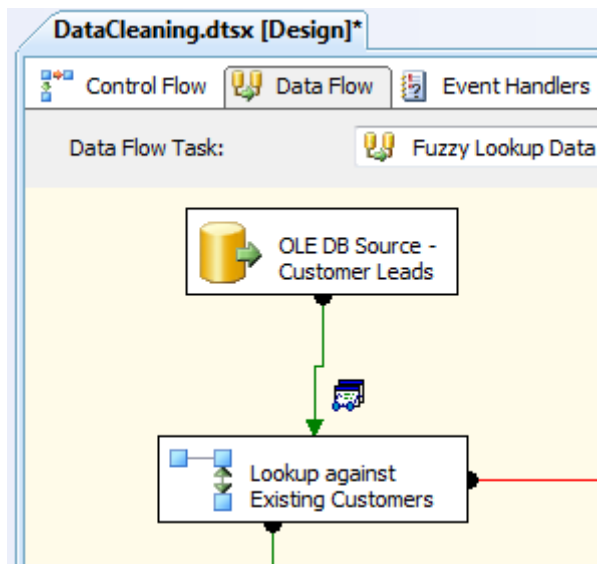


Figure 9–5. After you’ve added a data viewer to a data flow of an SSIS package, the package will show the small table-shaped icon (with glasses) next to the data flow path arrow where the data flow has been configured.

After you’ve added a data viewer, when you execute the package in BIDS, package execution will halt or break to display the data viewer. The associated data will be shown in a pop-up window. You can then manually continue package execution by clicking the small green triangle on the data viewer window. This process is a type of visual debugging. You can also detach and/or copy the data in the window to the clipboard. The transformation status execution will be paused (the components will be colored yellow) until you manually allow the package to continue by clicking the green triangle on the data viewer window. The data grid type data viewer for an executing package is shown in Figure 9–6.

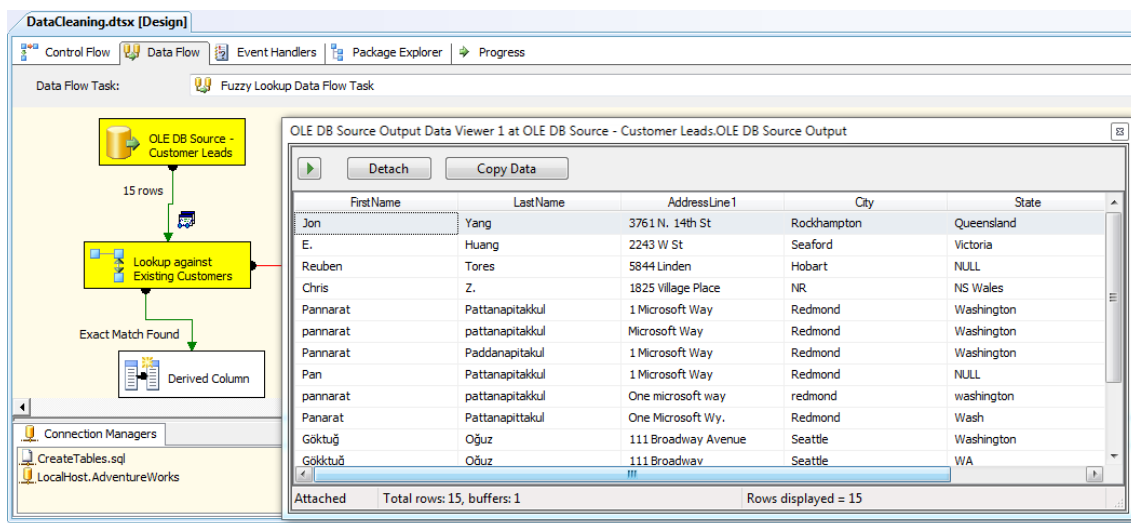


Figure 9–6. Executing a package with an associated data grid data viewer allows you to review the data as it flows through the pipeline.

Debugging SSIS Packages

You’ve already seen the first type of debugging now available in SSIS—using data viewers to debug data flows. An SSIS package breakpoint adds the ability for a package or an individual component to be paused when a certain condition (or event) fires. Additionally, you can configure breakpoints to fire when a certain condition fires at a precise number of times.

To add a breakpoint to a package, right-click the control flow design window and then click **Edit Breakpoints**. To add a breakpoint to a particular task, right-click the task and then click **Edit Breakpoints**. Breakpoints are set on one or more break conditions. A *break condition* is the name for an SSIS event. Figure 9–7 shows the list of available SSIS events.

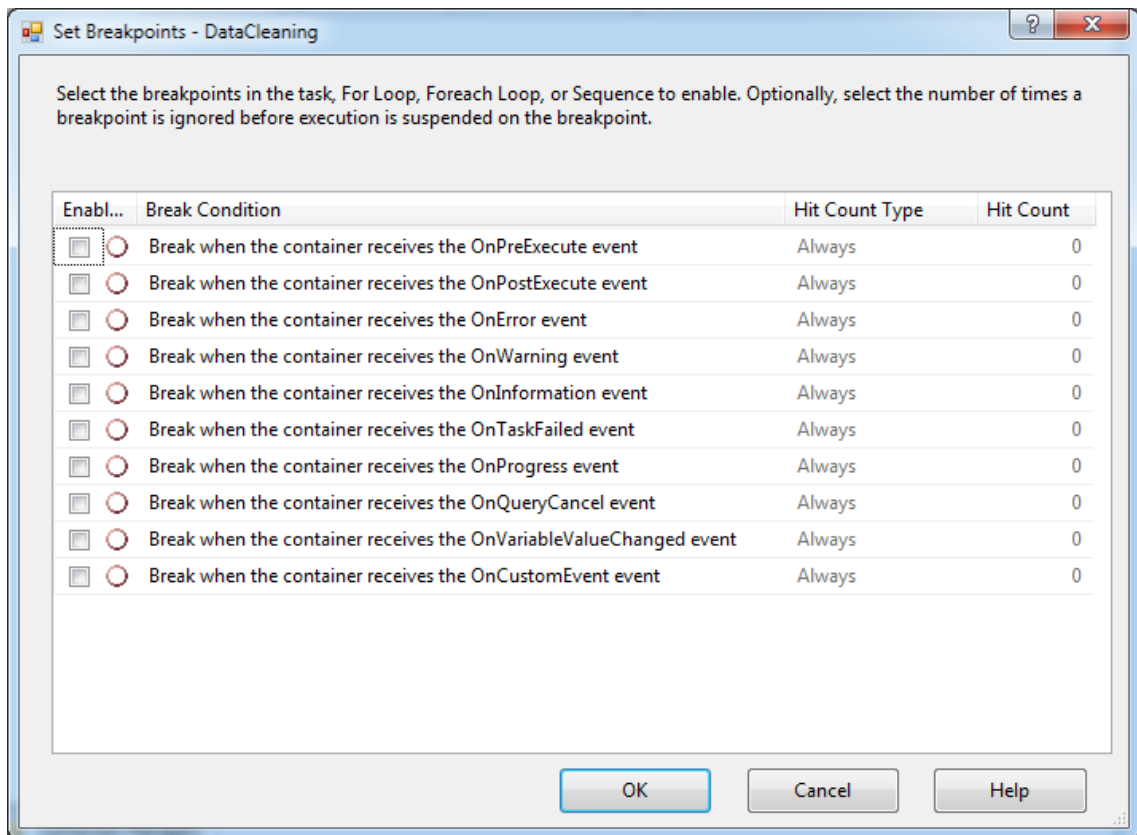


Figure 9–7. Setting breakpoints on a package or task for one or more break conditions.

For each break condition, you may also specify the Hit Count Type (choose from Always, Hit Count Equals, Hit Count Greater Than or Equal To, Hit Count Multiple) and/or Hit Count Value. After you’ve added your breakpoints, you can execute your package in BIDS and use all of the “classic” debugging tools of the Watch and Locals windows.

After you configure a breakpoint for a particular component, BIDS will add a red dot to that component on the design surface. When you execute a package that contains components with breakpoints configured, your package will show that the particular component has stopped at the configured breakpoint by adding a yellow arrow to the inside of the red dot that is placed on that component. Figure 9–8 shows the design surface for a task in break mode.

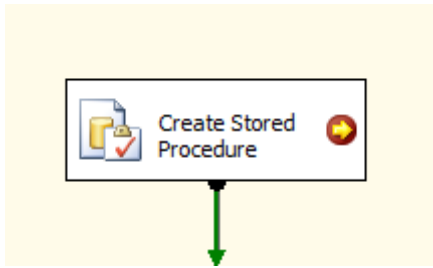


Figure 9-8. When an executing package halts at a breakpoint, the red dot shows a yellow arrow inside it to indicate that the package is in break mode and has halted at this task.

While in break mode, you can choose **Debug ► Windows ► Breakpoints** from the BIDS main menu to open a debug window to see even more information about your executing package. Figure 9-9 shows an example of the output available in the Locals window, which shows the current values of local variables for a particular task of a running package.

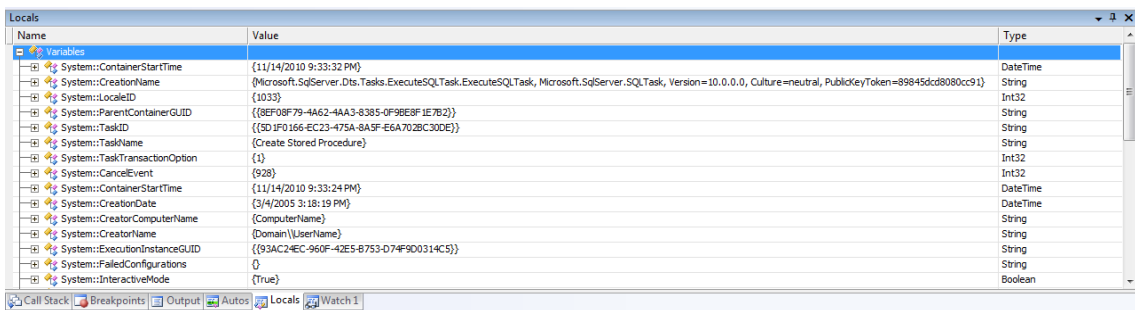


Figure 9-9. Viewing the Locals window for a particular task.

You can also add breakpoints to any script you write using the Control Flow Script task as well as by using the Data Flow Script component. You do this by right-clicking in the left (grey) margin next to the line of code where you want the application to break in the Script Design window. These debugging capabilities improve your productivity when implementing an ETL solution for your BI project, particularly as your SSIS packages increase in complexity. Also, if you use any scripts inside of your packages, you'll particularly appreciate the ability to work with breakpoints.

Most BI projects also require extensive attention to logging and error handling; both of these subjects are covered in the next section in detail.

Logging Execution Results

To access execution logging in SSIS, you choose **SSIS ► Logging** from the main BIDS menu, or you right-click the package design surface work area and then click **Logging**. You'll then be working with the **Configure SSIS Logs** dialog box. Your execution logging options include selecting which control flow task execution outputs you want to log and which log providers you want to use for logging. The log providers determine the method for logging and the destination for the log information. You have a choice of five different stock log providers. Figure 9-10 shows the **Configure SSIS Logs** dialog box and the five

destination options: a SQL Server table, a SQL Profiler trace file, a text file, the Windows event log, or an XML file.

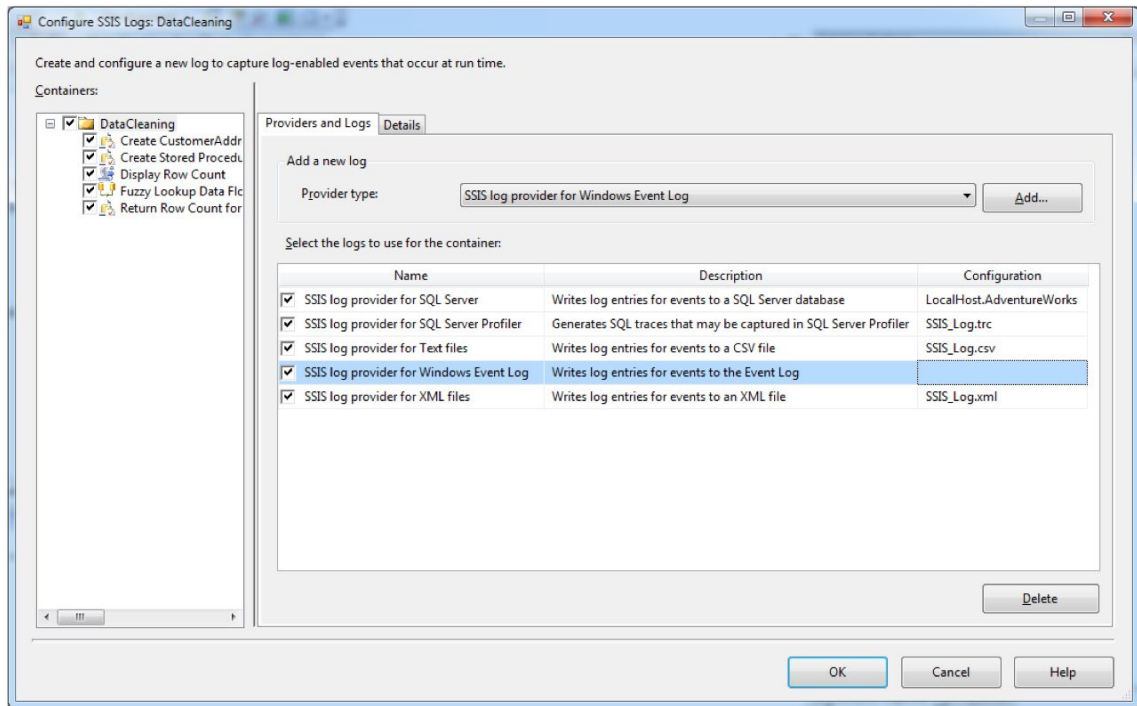


Figure 9–10. You can add log providers for your package execution logs in the *Configure SSIS Logs* dialog box. You also select which control tasks and which event outputs you want to capture.

After you've selected the control flow tasks and log locations, you then use the Details tab of the dialog box, as shown in Figure 9–11, to select the particular events that you are interested in logging. Figure 9–11 shows the selection of events you have to pick from. Note that Figure 9–11 shows one of the most commonly selected logging events, *OnTaskFailed*.

After logging is enabled, executing the package in BIDS generates all of the configured logs for you and allows you to view the logged information in the Log Events window. To view this window, choose **SSIS ► Log Events**. The Log Events window will open over the toolbox window by default. Figure 9–12 shows an example of this window for an executing package.

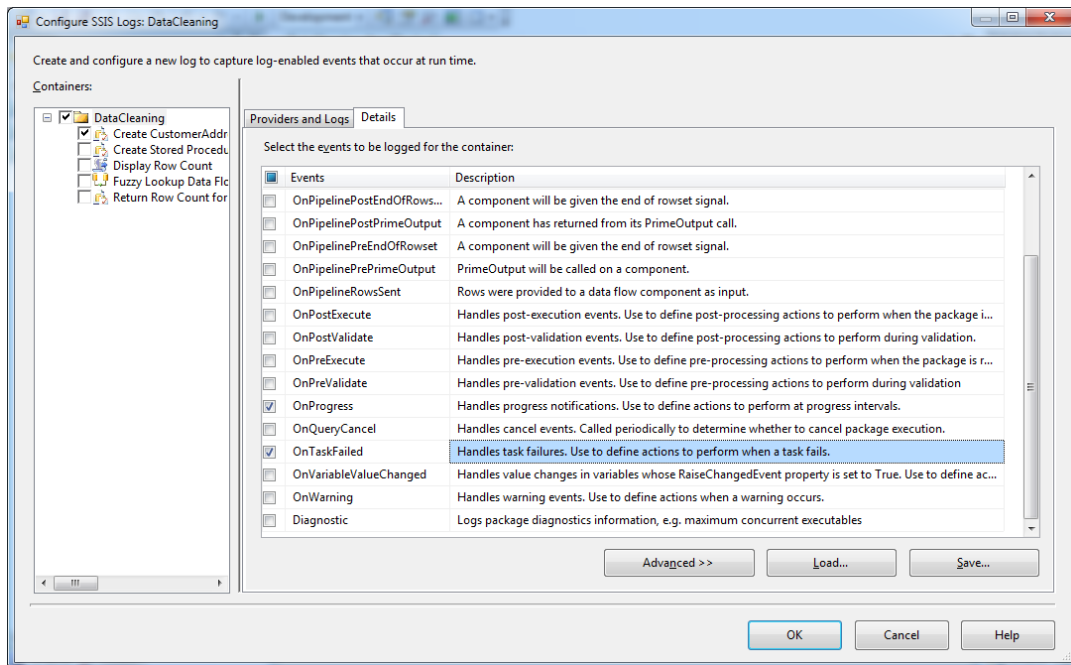


Figure 9-11. In the Details tab of the Configure SSIS Logs dialog box, you select the events that you are interested in capturing in your logs.

Figure 9-12 displays the 'Log Events' window, which provides a detailed view of the events captured during the execution of an SSIS package. The table below represents the data shown in the window.

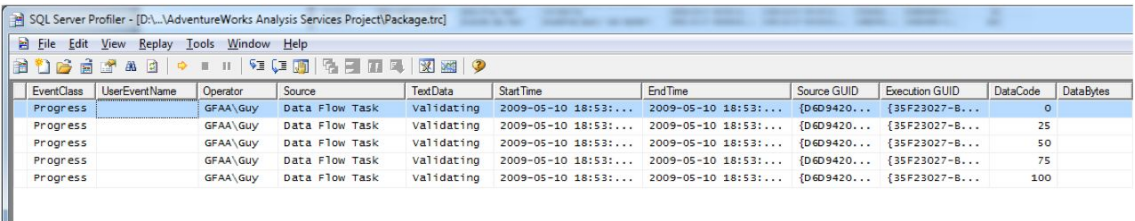
| Name | Computer | Operator | Source Name | Source GUID | Execution GUID | Message | Start Time | End Time |
|----------------|----------|----------|-----------------------------|-----------------|------------------|---|------------------------|------------------------|
| OnPostValidate | GFAA | GFAA\... | DataCleaning | {8EF08F79-4A... | {68891F64-3D4... | | 11/14/2010 10:23:56 PM | 11/14/2010 10:23:56 PM |
| PackageStart | GFAA | GFAA\... | DataCleaning | {8EF08F79-4A... | {68891F64-3D4... | Beginning of package execution. | 11/14/2010 10:23:56 PM | 11/14/2010 10:23:56 PM |
| OnPreExecute | GFAA | GFAA\... | DataCleaning | {8EF08F79-4A... | {68891F64-3D4... | | 11/14/2010 10:23:56 PM | 11/14/2010 10:23:56 PM |
| OnPreExecute | GFAA | GFAA\... | Fuzzy Lookup Data Flow Task | {4D509206-63... | {68891F64-3D4... | | 11/14/2010 10:23:57 PM | 11/14/2010 10:23:57 PM |
| OnPreValidate | GFAA | GFAA\... | Fuzzy Lookup Data Flow Task | {4D509206-63... | {68891F64-3D4... | | 11/14/2010 10:23:57 PM | 11/14/2010 10:23:57 PM |
| OnInformation | GFAA | GFAA\... | Fuzzy Lookup Data Flow Task | {4D509206-63... | {68891F64-3D4... | Validation phase is beginning. | 11/14/2010 10:23:57 PM | 11/14/2010 10:23:57 PM |
| OnInformation | GFAA | GFAA\... | DataCleaning | {8EF08F79-4A... | {68891F64-3D4... | Validation phase is beginning. | 11/14/2010 10:23:57 PM | 11/14/2010 10:23:57 PM |
| OnPostValidate | GFAA | GFAA\... | Fuzzy Lookup Data Flow Task | {4D509206-63... | {68891F64-3D4... | | 11/14/2010 10:23:57 PM | 11/14/2010 10:23:57 PM |
| OnInformation | GFAA | GFAA\... | Fuzzy Lookup Data Flow Task | {4D509206-63... | {68891F64-3D4... | Prepare for Execute phase is beginning. | 11/14/2010 10:23:57 PM | 11/14/2010 10:23:57 PM |
| OnInformation | GFAA | GFAA\... | DataCleaning | {8EF08F79-4A... | {68891F64-3D4... | Prepare for Execute phase is beginning. | 11/14/2010 10:23:57 PM | 11/14/2010 10:23:57 PM |
| OnInformation | GFAA | GFAA\... | Fuzzy Lookup Data Flow Task | {4D509206-63... | {68891F64-3D4... | Pre-Execute phase is beginning. | 11/14/2010 10:23:57 PM | 11/14/2010 10:23:57 PM |
| OnInformation | GFAA | GFAA\... | DataCleaning | {8EF08F79-4A... | {68891F64-3D4... | Pre-Execute phase is beginning. | 11/14/2010 10:23:57 PM | 11/14/2010 10:23:57 PM |
| OnInformation | GFAA | GFAA\... | Fuzzy Lookup Data Flow Task | {4D509206-63... | {68891F64-3D4... | The component "Lookup against Existing Cus... | 11/14/2010 10:23:57 PM | 11/14/2010 10:23:57 PM |
| OnInformation | GFAA | GFAA\... | DataCleaning | {8EF08F79-4A... | {68891F64-3D4... | The component "Lookup against Existing Cus... | 11/14/2010 10:23:57 PM | 11/14/2010 10:23:57 PM |
| OnInformation | GFAA | GFAA\... | Fuzzy Lookup Data Flow Task | {4D509206-63... | {68891F64-3D4... | The Maximum insert commit size property of ... | 11/14/2010 10:23:59 PM | 11/14/2010 10:23:59 PM |
| OnInformation | GFAA | GFAA\... | DataCleaning | {8EF08F79-4A... | {68891F64-3D4... | The Maximum insert commit size property of ... | 11/14/2010 10:23:59 PM | 11/14/2010 10:23:59 PM |
| OnInformation | GFAA | GFAA\... | Fuzzy Lookup Data Flow Task | {4D509206-63... | {68891F64-3D4... | The Maximum insert commit size property of ... | 11/14/2010 10:23:59 PM | 11/14/2010 10:23:59 PM |
| OnInformation | GFAA | GFAA\... | DataCleaning | {8EF08F79-4A... | {68891F64-3D4... | The Maximum insert commit size property of ... | 11/14/2010 10:23:59 PM | 11/14/2010 10:23:59 PM |
| OnInformation | GFAA | GFAA\... | Fuzzy Lookup Data Flow Task | {4D509206-63... | {68891F64-3D4... | The Maximum insert commit size property of ... | 11/14/2010 10:23:59 PM | 11/14/2010 10:23:59 PM |
| OnInformation | GFAA | GFAA\... | DataCleaning | {8EF08F79-4A... | {68891F64-3D4... | The Maximum insert commit size property of ... | 11/14/2010 10:23:59 PM | 11/14/2010 10:23:59 PM |
| OnInformation | GFAA | GFAA\... | Fuzzy Lookup Data Flow Task | {4D509206-63... | {68891F64-3D4... | Execute phase is beginning. | 11/14/2010 10:23:59 PM | 11/14/2010 10:23:59 PM |
| OnInformation | GFAA | GFAA\... | DataCleaning | {8EF08F79-4A... | {68891F64-3D4... | Execute phase is beginning. | 11/14/2010 10:23:59 PM | 11/14/2010 10:23:59 PM |
| OnInformation | GFAA | GFAA\... | Fuzzy Lookup Data Flow Task | {4D509206-63... | {68891F64-3D4... | Rows processed by Fuzzy Lookup : 0 | 11/14/2010 10:23:59 PM | 11/14/2010 10:23:59 PM |
| OnInformation | GFAA | GFAA\... | Fuzzy Lookup Data Flow Task | {4D509206-63... | {68891F64-3D4... | Rows processed by Fuzzy Lookup : 14 | 11/14/2010 10:23:59 PM | 11/14/2010 10:23:59 PM |
| OnInformation | GFAA | GFAA\... | Fuzzy Lookup Data Flow Task | {4D509206-63... | {68891F64-3D4... | The final commit for the data insertion in "co... | 11/14/2010 10:23:59 PM | 11/14/2010 10:23:59 PM |
| OnInformation | GFAA | GFAA\... | DataCleaning | {8EF08F79-4A... | {68891F64-3D4... | The final commit for the data insertion in "co... | 11/14/2010 10:23:59 PM | 11/14/2010 10:23:59 PM |
| OnInformation | GFAA | GFAA\... | Fuzzy Lookup Data Flow Task | {4D509206-63... | {68891F64-3D4... | The final commit for the data insertion in "co... | 11/14/2010 10:23:59 PM | 11/14/2010 10:23:59 PM |

Figure 9-12. You can view the logged events in an executing package in BIDS.

Tip To see more detail about any listed event in the Log Events window, just double-click the event to open a details window in BIDS.

Our favorite log provider is the SQL Server Profiler log provider. It outputs to a SQL Server Profiler trace file, which you can open (the file type is .trc) in Profiler; just double-click the file. Once inside Profiler, you can view the file and, importantly, further analyze the file using the associated tools, such as the Database Engine Tuning Advisor (DETA).

For example, you could use the SQL Server Profiler Log Provider in an SSIS package to save events to a trace file. You can subsequently use that trace file as input to DETA. You could expect to see results from the DETA, including optimizing SSIS data flows by adding, removing, or changing indexes on SQL Server source or destination data, for example. Figure 9–13 shows a log trace file opened in Profiler. Of course, tools like DETA are designed to work only with SQL Server source and destination data.



| EventClass | UserEventName | Operator | Source | TextData | StartTime | EndTime | Source GUID | Execution GUID | DataCode | DataBytes |
|------------|---------------|----------|----------------|------------|----------------------|----------------------|--------------|-----------------|----------|-----------|
| Progress | | GFAA\Guy | Data Flow Task | Validating | 2009-05-10 18:53:... | 2009-05-10 18:53:... | {D6D9420...} | {35F23027-B...} | 0 | |
| Progress | | GFAA\Guy | Data Flow Task | Validating | 2009-05-10 18:53:... | 2009-05-10 18:53:... | {D6D9420...} | {35F23027-B...} | 25 | |
| Progress | | GFAA\Guy | Data Flow Task | Validating | 2009-05-10 18:53:... | 2009-05-10 18:53:... | {D6D9420...} | {35F23027-B...} | 50 | |
| Progress | | GFAA\Guy | Data Flow Task | Validating | 2009-05-10 18:53:... | 2009-05-10 18:53:... | {D6D9420...} | {35F23027-B...} | 75 | |
| Progress | | GFAA\Guy | Data Flow Task | Validating | 2009-05-10 18:53:... | 2009-05-10 18:53:... | {D6D9420...} | {35F23027-B...} | 100 | |

Figure 9–13. A SQL Server Profiler trace (or .trc) file

Whatever your business requirements for logging execution results in your SSIS packages, SSIS has the flexibility to meet your needs. At this point, we need to remind you about the data flow Audit task. It is quite common in BI solutions for SSIS logging to include requirements regarding logging data lineage. The Audit transformation discussed in Chapter 8 allows you to capture selected types of information during the execution of your SSIS package.

Error Handling

By default, one error is permitted in an SSIS package or control flow task. If this default is reached, either the control flow task or the entire package will halt execution and return failure. You also can configure several additional properties that control the error behavior. Following is a list and brief explanation (all are shown in Figure 9–14 in the task Properties dialog box as well) of each of these six properties:

- **FailPackageOnFailure:** This means to fail the entire package on the first task or transformation failure.
- **FailParentOnFailure:** This means to fail the parent package or task if this task (the child package or task) fails.
- **ForcedExecutionValue:** This means return the actual result of the **ForceExecutionValue** for the package (if that value has been set to True).
- **ForcedExecutionValueType:** This is the data type of the **ForcedExecutionValue** property.

- **ForceExecutionResult**: This means to force a container or task to return a specific result: Success, Failure, or Completion.
- **ForceExecutionValue**: This means to force, or always try to, execute the package; the default is False.

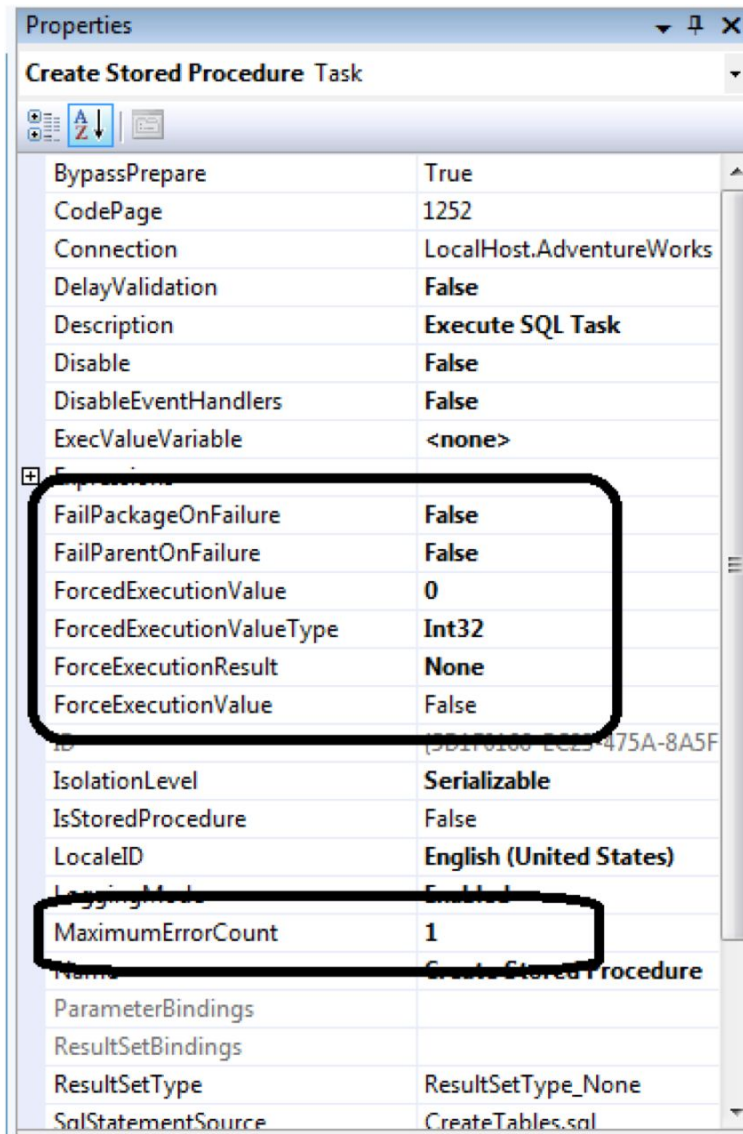


Figure 9-14. By configuring the *MaximumErrorCount* and other properties that affect error handling, you have enhanced control over error-handling results in your SSIS packages.

In addition to specifying how you want a package's or a particular control flow task's errors handled, you also can configure error handling for most data flows. This is done by right-clicking the data source or destination icon on the Data Flow design surface, clicking **Edit** on the shortcut menu, and then clicking **Error Output** in the left pane of the dialog box. This custom configuration is available for OLE DB, Excel, flat file, and XML data flow sources.

This selection brings up a dialog box (see Figure 9–15) that allows you to configure error output for each column in the data flow source or destination item. For each column, you have three choices for errors or truncations (truncations mean conversions that could result in data loss due to narrowing of the destination data type, for example, using a ten-character string source and a two-character string destination): **Ignore Failure**, **Redirect Row**, and **Fail Component**. Following are more complete descriptions of each option:

- **Ignore Failure:** The error (or truncation) is ignored, and the row is sent to the output of the transformation or source.
- **Redirect Row:** The error (or truncation) row is sent to the error output of the source, transformation, or destination.
- **Fail Component:** The data flow task will fail whenever an error or a truncation occurs. Fail Component is the default value.

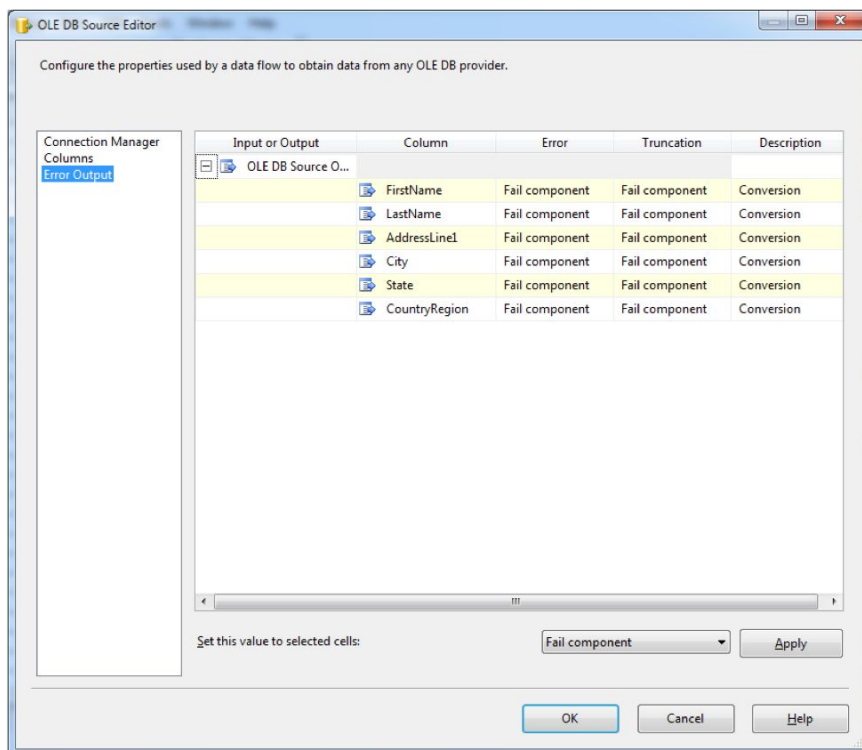


Figure 9–15. The error output dialog box of several data flow source and destination types allows you to specify the error behavior of your package at the level of individual column errors.

In addition to configuring logging and error handling, you can add control flows in response to various event handlers firing in your packages via the BIDS interface. We'll cover the details of doing this in the next section.

Event Handlers

SSIS includes an easy method for you to use from inside BIDS that allows you to add custom control flows to your SSIS packages as a result of package or control flow task events firing. To do this, click the Event Handlers tab of the SSIS BIDS designer. You'll see the Executable and Event handler drop-down lists. In the Executable list, select either the entire package or a specific control flow task. In the Event handler list, select the particular event for which you want to design a control flow. An example of creating a simple control flow for a data flow task using the `OnProgress` event is shown in Figure 9–16.

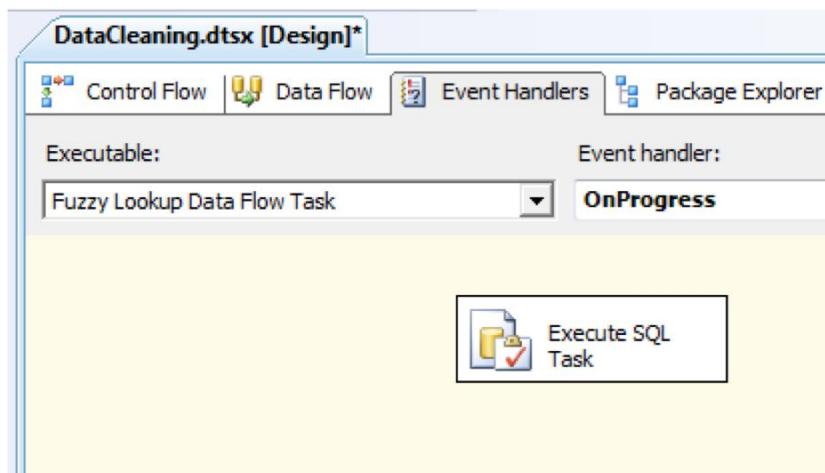


Figure 9–16. The Event Handlers tab of the SSIS designer allows you to easily create control flows that are associated with a particular event handler for the entire package or for a particular event handler for a control flow task.

After you've created a control flow for a particular event handler, that event name text is shown in bold on the drop-down list called "Event handler" on the right side of the BIDS design surface. The most commonly used event handlers are `OnError` and `OnTaskFailed`. The events available in the SSIS event handler design area are as follows:

- `OnError` is raised by an executable when an error occurs.
- `OnExecStatusChanged` is raised by an executable when its execution status changes.
- `OnInformation` is raised during validation or execution of an executable and displays only information, not errors or warnings.
- `OnPostExecute` is raised by an executable after it has completed running.
- `OnPostValidate` is raised by an executable after it has completed validation.

- `OnPreExecute` is raised by an executable before it runs.
- `OnPreValidate` is raised by an executable when its validation starts.
- `OnProgress` is raised by an executable when it makes measurable progress.
- `OnQueryCancel` is raised by an executable to decide whether it should stop.
- `OnTaskFailed` is raised by a task when it fails.
- `OnVariableValueChanged` is raised by an executable when the value of an associated variable changes.
- `OnWarning` is raised by an executable when a warning runs.

As with many other aspects of SSIS, you can also create event handlers programatically. BOL has complete descriptions and sample code if you want to explore this possibility.

Deploying SSIS Packages

When you've completed implementing your SSIS package (or packages) in BIDS, you'll next want to deploy the package to a testing and, eventually, into a production environment. To start this process, you must first validate the SSIS packages you've created in BIDS by working with the SSIS project menu. When you right-click the SSIS project name in the Solution Explorer, the options available are **Build** and **Rebuild**. What exactly do these options do?

First, you need to understand what exactly you are creating in the SSIS designer. Although you are working visually, you are creating an executable file. The code that you are writing when you create an SSIS package is a dialect of XML. If you want to view this XML, right-click any `.dtsx` package in the Solution Explorer and then click **View Code**. You'll see something similar to Figure 9–17.

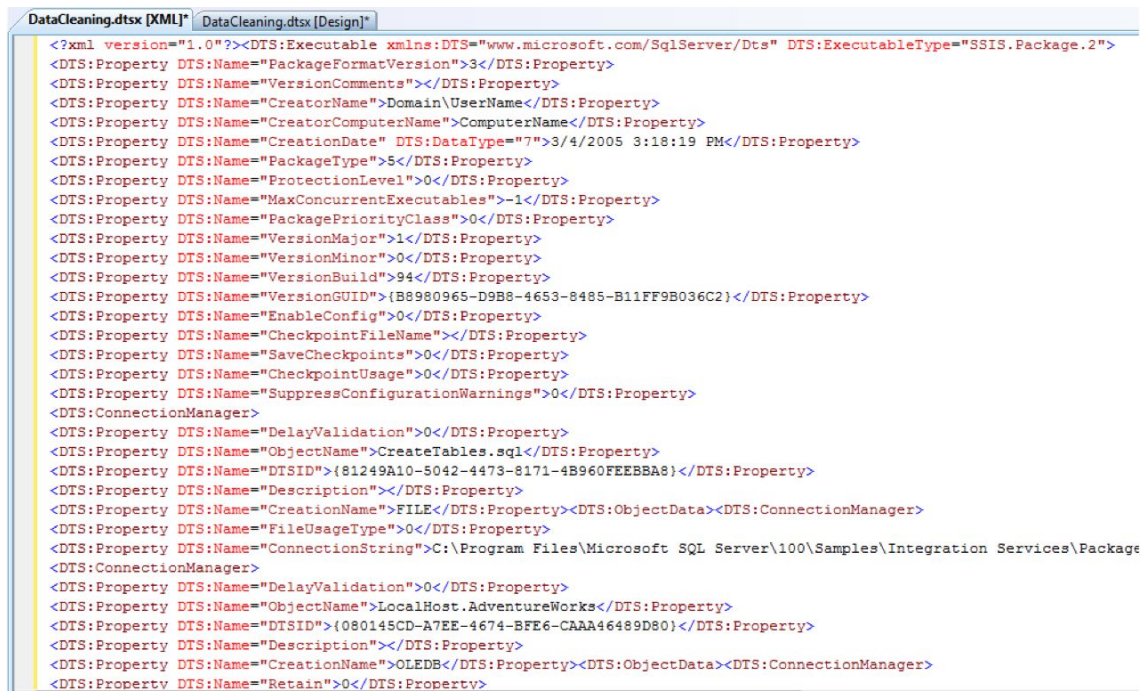


Figure 9–17. You are actually creating XML metadata when you visually design .dtsx packages using the SSIS template in BIDS.

Note that you’re validating either the entire output or the changed output of each XML file for each .dtsx package against the associated specialized XSD schema when you select the Build or Rebuild option in Solution Explorer. If there are validation errors, BIDS will show them to you in the Error List pane so that you can resolve those errors prior to package deployment. After your package or packages have been built successfully, your next step is to deploy them to a testing or production server. Let’s look at a couple of different options for deploying your packages.

SSIS Package Deployment Options

Although you could deploy and run your SSIS packages manually, you’ll rarely manage and execute your SSIS packages this way unless you are in a development and testing environment. Rather, you’ll likely prefer to automate and run your packages in most production BI environments.

BIDS includes a Package Installation Wizard that can simplify deployment of your SSIS packages. To deploy your package using the SSIS Package Installation Wizard, you first set the package properties by right-clicking the SSIS project name in the Solution Explorer and selecting Properties. Set the CreateDeploymentUtility property to True, and then build your project. This will create an SSIS manifest file at the location configured in the DeploymentOutputPath property, as shown in Figure 9–18.

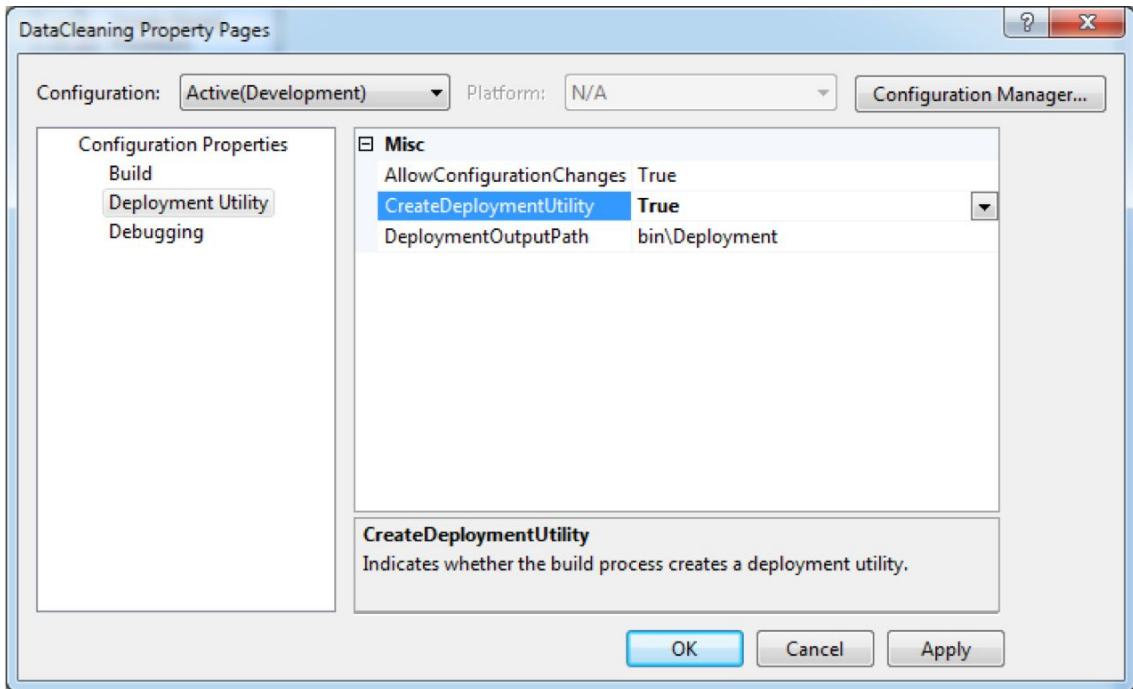


Figure 9–18. Configure the `CreateDeploymentUtility` property to generate an SSIS manifest file.

Next, you locate the manifest file, which has the name format of `<ProjectName>.SSISDeploymentManifest` and contains your package metadata in an XML format. Double-click your manifest file to launch the Package Installation Wizard. Using the wizard, you can select whether to have your packages deployed to the file system or SQL Server. You can also optionally validate packages after installation. The Deploy SSIS Packages section of the Package Installation Wizard is shown in Figure 9–19. If you choose to validate your package, a Package Validation section in the wizard will list invalid or inefficient task configurations in your package.

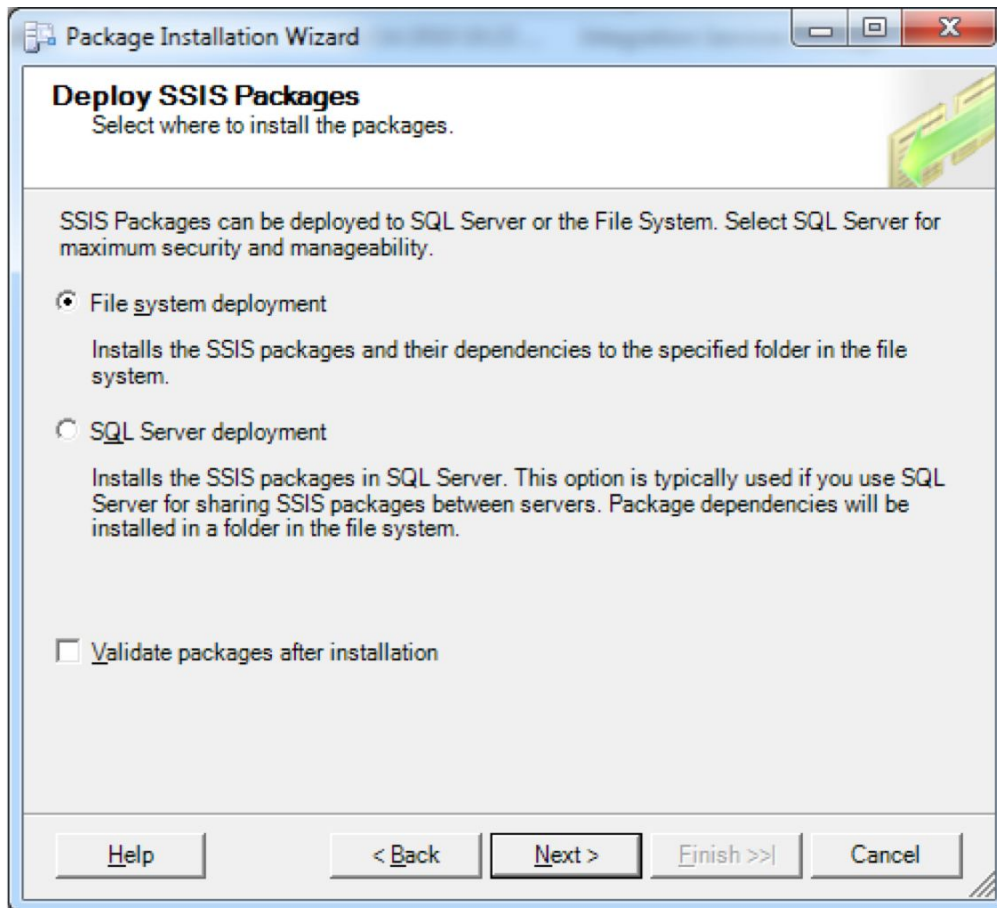


Figure 9–19. You can select the destination for installation for your SSIS packages using the Package Installation Wizard.

Another option for package deployment is to use the command-line tool `dtutil.exe` to copy, move, delete, or take other actions on your SSIS package. You will often choose this option in production because you can script SSIS package deployment with this tool. The syntax for this command is shown in Figure 9–20.

Still another option for deployment is to simply import the packages into the desired location. You do this using SSMS by right-clicking the node in the SSIS tree (file, MSDB, and so on) and then clicking Import Package. Figure 9–21 in the next section shows the SSMS SSIS interface.

After you've deployed your packages to one of the three possible locations—SQL Server MSDB (Multisource Database), default file system, or specified location on the file system—you have a couple of choices about how you'll run them.

```

Administrator: Command Prompt
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Guy>dtutil /?
Microsoft (R) SQL Server SSIS Package Utilities
Version 10.50.1600.1 for 32-bit
Copyright (C) Microsoft Corporation 2010. All rights reserved.

Usage: DTUtil /option [value] [/option [value]] ...
Options are case-insensitive.
A hyphen (-) may be used in place of a forward slash (/).
The vertical bar (|) is the OR operator and is used to list possible values.
For extended help use /help with an option. For example: DTUtil /help Copy

/Cl[copy]           {SQL | FILE | DTS};Path
/Dec[rypt]         Password
/Del[ete]
/Des[t]Passw[ord]   Password
/Des[t]Serv[er]     Server
/Des[t]U[s]er       User name
/D[TS]             PackagePath
/D[ump]            Process ID
/En[crypt]         {SQL | FILE | DTS};Path;ProtectionLevel[;Password]
/Ex[ists]
/FC[reate]         {SQL | DTS};ParentFolderPath;NewFolderName
/FDe[lete]         {SQL | DTS};ParentFolderPath;FolderName
/FDi[rectory]      {SQL | DTS};FolderPath[;S1]
/FE[xists]         {SQL | DTS};FolderPath
/FR[ename]        {SQL | DTS};ParentFolderPath;OldFolderName;NewFolderName
/Fi[le]           Filespec
/H[elp]           [Option]
/ID[Regenerate]
/M[ove]           {SQL | FILE | DTS};Path
/Q[uiet]
/R[emark]         [Text]
/Si[gn]          {SQL | FILE | DTS};Path;Hash
/SourceP[assword] Password
/SourceS[erver]   Server
/SourceU[s]er     User name
/SQ[L]           PackagePath

C:\Users\Guy>

```

Figure 9–20. The command-line tool `dtutil.exe` allows you to script actions on your SSIS packages.

SSIS Package Execution Options

A first option for package execution is to use SSMS. This method of SSIS package execution is primarily used for development and testing. The second and third options are to use either a GUI package execution tool (`dtexecui.exe`), which runs outside of SSMS, or to use a command-line tool, which also runs outside of SSMS (`dtexec.exe`). The latter two are the preferred methods in production environments.

To run SSIS packages from within SSMS, you'll first connect to SSIS from the SSMS Object Explorer. Once connected, you can view a list of packages grouped by storage location and then configure execution properties for any package you want to execute. You can then right-click any package and choose **Run Package** to execute it via the **Execute Package Utility**. Figure 9–21 shows a connection to SSIS in SSMS. Note that you can view packages in two locations: **File System** or **MSDB** nodes in Object Explorer. The **Execute Package Utility** contains a large number of configurable runtime properties for

your packages. For example, Figure 9–22 shows the reporting options for packages. The reporting options include selecting the level of verbosity for logging console events and the type of information to capture in console logs.

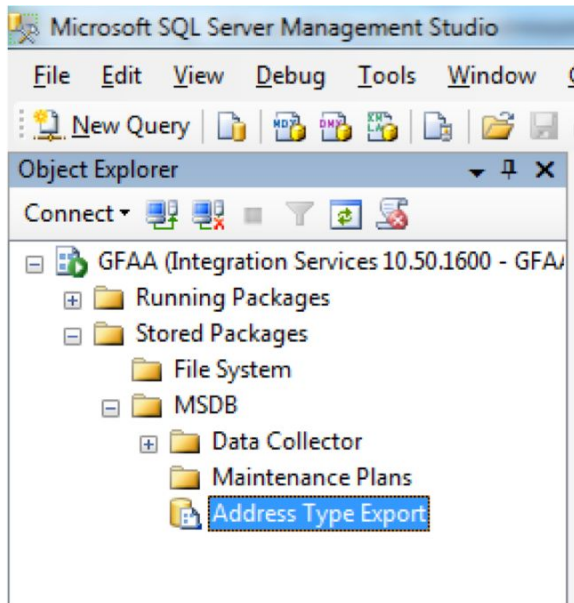


Figure 9–21. You can view SSIS packages grouped by storage location, configure execution properties, and execute packages after connecting to SSIS in SSMS.

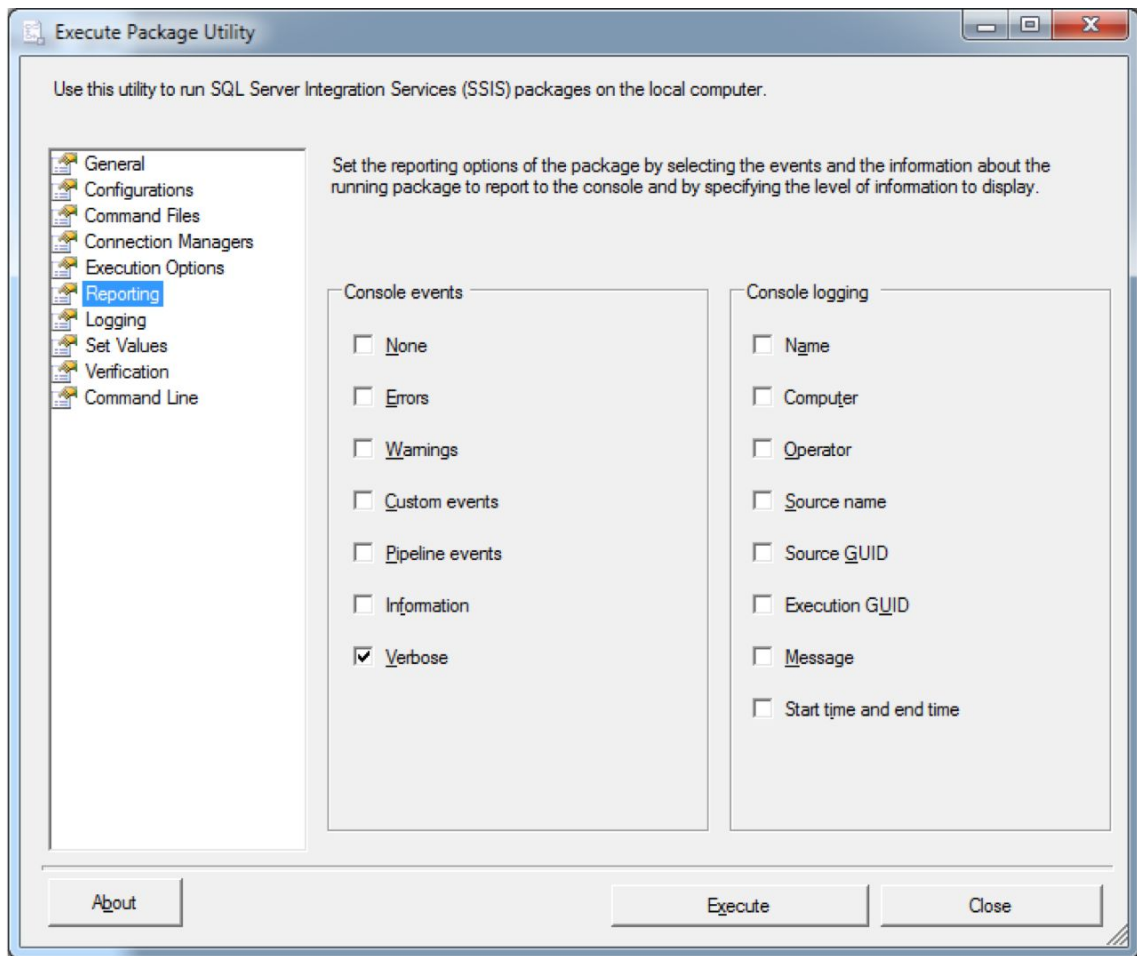


Figure 9–22. The *Execute Package Utility* dialog box in SSMS allows you to configure many types of reporting events and logs for each executing package.

Continuing with another example, Figure 9–23 shows the “Log providers” configuration area. In production, business requirements often drive logging requirements for your production SSIS packages. Sometimes, this will be more formal than others; that is, you may be subject to regulatory requirements, such as SOX or HIPAA.

After you’ve executed the associated package at least once, one way to view the logs that you have configured in the procedure described previously is to right-click the top node (SSIS) in SSMS and then click View Logs. You’ll see the output similar to what is shown in Figure 9–24.

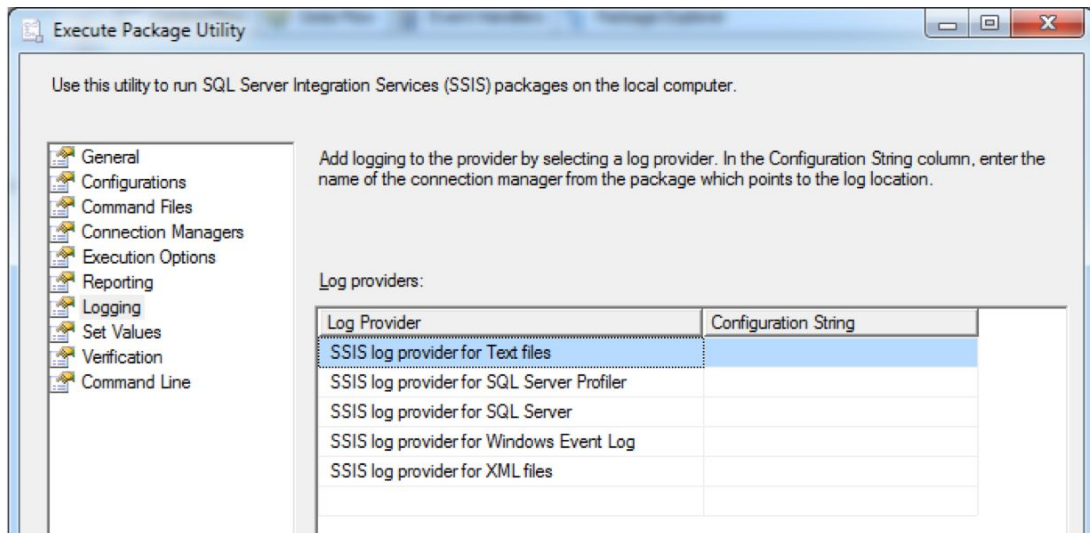


Figure 9-23. By using the logging section of the *Execute Package Utility* dialog box, you can associate one or more log providers with the package.

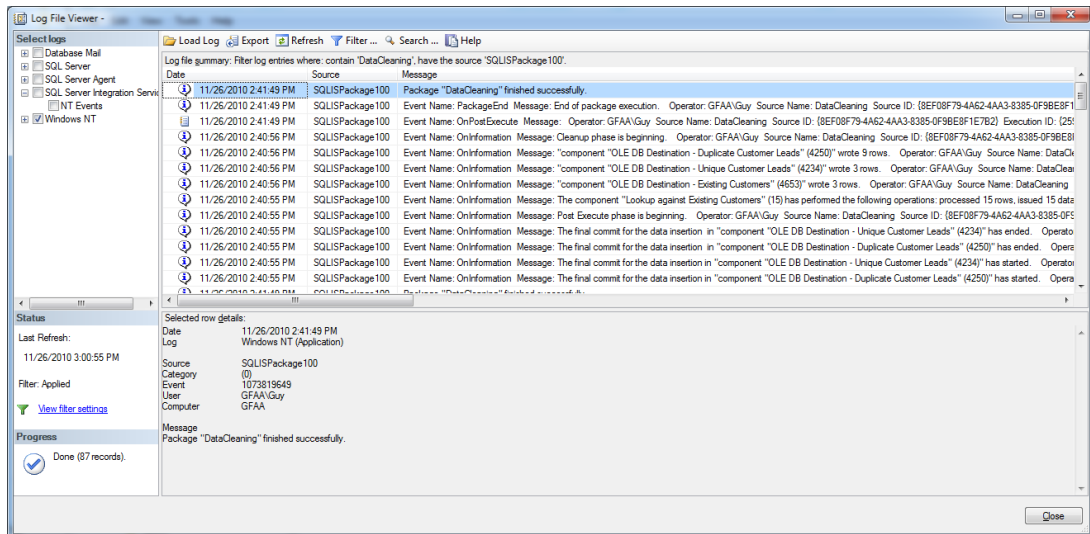


Figure 9-24. You can view any logs that you've configured via the *Execute Package Utility* in SSMS.

As you may have noticed, you can configure additional runtime package properties by using the *Execute Package Utility* dialog box. In production systems, the most commonly configured properties belong to the Connection Managers, Reporting, and Logging categories of the *Execute Package Utility* properties.

■ **Tip** After you move your BI solution to production, it is typical to execute SSIS packages by using scripts. These scripts are most often run in the context of `dtexec.exe`. You can run these scripts from the command line, or you can use SQL Server Agent jobs to schedule these executables. In production, you'll often use `dtexec.exe` (rather than the SSMS GUI), because you can more easily automate package execution via scripts using the command-line tool.

Of course, another important consideration with deployment of packages is security. We'll look at the SSIS security model in a bit more depth in the next section of this chapter.

SSIS Package Security

The security options you have to consider for your SSIS packages depend on where you choose to store them. You have three options: within a SQL Server instance in the MSDB database, in the default location on the file system, or in a named location on the file system. Choosing to store your packages in MSDB is the most typical scenario for production solutions. In this case, you can use the built-in SQL Server security roles to limit access to your packages:

- `db_dtsadmin`: This role has full administrative rights on all SSIS packages stored in MSDB.
- `db_dtsltduser`: This role can view, execute, change, or export only its own packages and can import any packages.
- `db_dtsoperator`: This role can view, execute, and export packages.

You could, of course, also create custom roles for the MSDB if your business requirements call for more granular levels of control over packages.

If you choose to store packages as `.dtsx` files on the file system, you will assign NTFS permissions to those storage locations. Another consideration when assigning NTFS permissions is assigning appropriate permissions to external files used by SSIS packages. Some examples of possible external files that could be used by SSIS packages are listed here:

- *Data source files*: Excel, Text, XML
- *Configuration files*: XML, Text, SQL scripts
- *Package execution log files*: Text, XML, trace files

Another aspect of securing your SSIS packages involves configuring the `ProtectionLevel` property of the package. Here, you'll specify whether sensitive data should be protected by encryption or whether the sensitive data should be removed before saving the package by choosing one of the following options:

- *DontSaveSensitive*: Removes any data that is tagged as "sensitive" from the package when it is saved. When the package is opened, after having been saved with this protection level, the end user must provide the values for the properties that were marked sensitive.

- *EncryptSensitiveWithUserKey*: Encrypts sensitive package data with a key from the user profile. Only the same user with the same profile can run the package. This is the default option for SSIS packages.
- *EncryptSensitiveWithPassword*: Encrypts sensitive package data with a user-supplied password. To run the package, the end user must supply the password.
- *EncryptAllWithPassword*: Encrypts all package data with a user-supplied password. To run the package, the end user must supply the password.
- *EncryptAllWithKey*: Encrypts all package data with a key from the user profile. Only the same user with the same profile can run the package.
- *ServerStorage*: Protects the package based on SQL server database roles. This option is only available for packages stored in MSDB.

Note You cannot control which property values are considered sensitive. SSIS sensitive information includes the password portion of connection strings, some XML node information, and certain other properties; according to BOL, “The marking of variables (as sensitive) is controlled by Integration Services.”

Although you set the initial package protection level when designing the package in BIDS, you can update this setting after the package has been deployed. Some of the *ProtectionLevel* options require that you associate a password with your package. You do this by configuring the *PackagePassword* property.

The final security option you can consider is whether you need to guarantee the integrity of packages, or to more formally prevent tampering with the contents of packages. If this is a requirement for your project, you’ll sign your packages with certificates. This option requires associating a certificate that has been created for code-signing purposes with your package. To do this in BIDS, you choose SSIS ► Digital Signing, and then click the Sign button. A Select Certificate dialog box appears. You select the appropriate certificate and click OK twice.

Placing Checkpoints

Checkpointing refers to placing a marker, or checkpoint, in the control flow so that, if a package fails, you can restart the package from the point of failure. Execution information is saved into a checkpoint file. Figure 9–25 shows this property option. If you choose to use checkpointing in SSIS packages, you’ll want to include this capability in the initial package design of all affected packages.

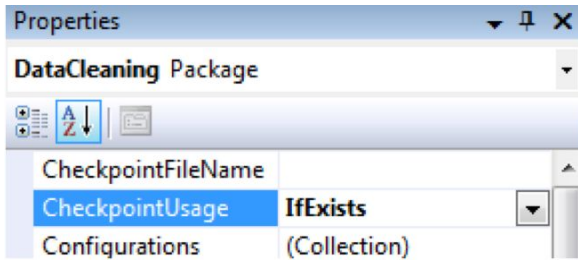


Figure 9–25. To enable checkpointing in a package, you must set the *CheckpointUsage* property to either *IfExists* or *Always* and then set three other properties.

To implement checkpointing in your SSIS package, you must configure four properties:

- **CheckpointUsage:** This must be set to *IfExists* or *Always*.
- **CheckpointFileName:** This is the path to the checkpoint file.
- **SaveCheckpoints:** This must be set to *True* to enable package checkpoints.
- **FailPackageOnFailure:** This must be set to *True* to enable package checkpoints.

Caution Checkpointing only allows package restart for failures that occur during the execution of control flow tasks. You must implement other error-handling and recovery methods to recover from data flow transformation failures. Also, if the checkpoint file is deleted or altered, it cannot be used for recovery.

One method of recovering from data flow task failures is to use transactions. We'll give you a brief introduction to this advanced technique in the next section.

Using Transactions in SSIS Packages

SSIS packages support the grouping of tasks into transactions by using one of two methods. You can either use the Distributed Transaction Coordinator (DTC), or you can use the transactions that are built in to your relational database management system (RDMS), such as SQL Server or Oracle. The primary factor to consider is the span of the tasks.

If your business needs call for transactions that must go across multiple data sources or connections, you must use DTC transactions. To use DTC transactions, you must ensure that the DTC service is installed and running. You then configure the transaction option at a package or task level by setting the *TransactionOption* property to either *Required* or *Supported*. You'll also use sequence containers within SSIS as transactional boundaries.

If all of your transactions will take place within the same RDMS, such as SQL Server, you can use RDMS-native transactions. For our example, we'll discuss SQL Server transactions.

The *TransactionOption* property for an SSIS package is shown in Figure 9–26. You also may configure the *IsolationLevel* property for the transaction. The *IsolationLevel* property setting affects the locking behavior of the involved data sources during the execution of the transaction.

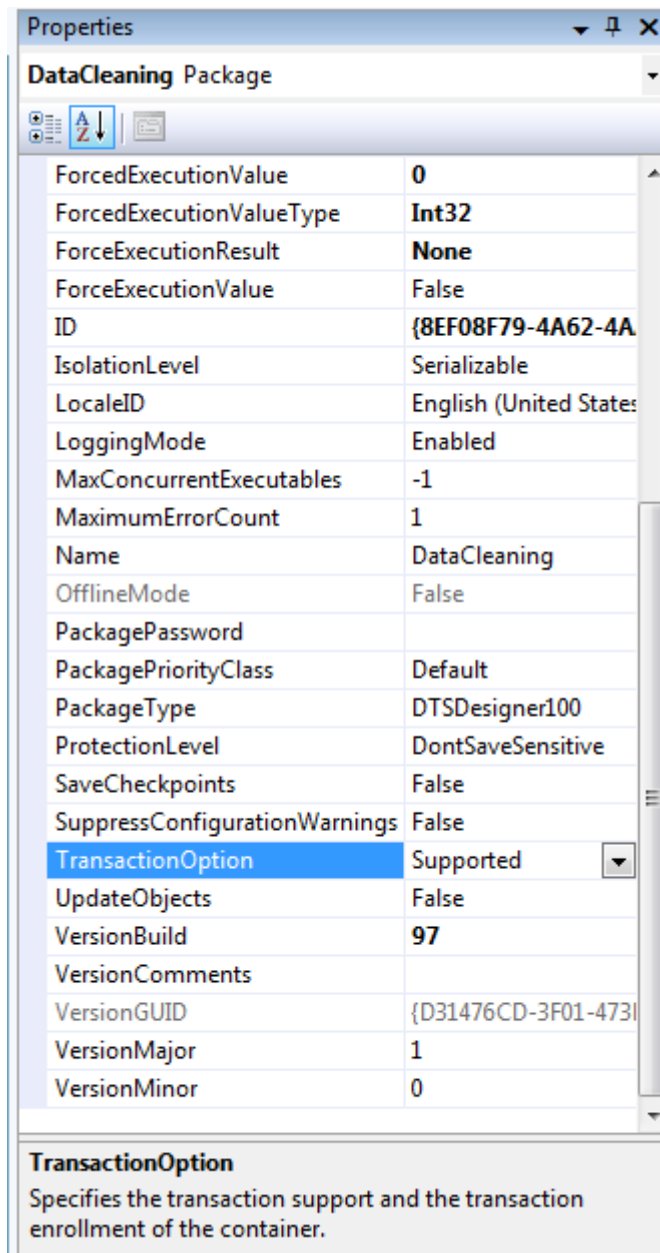


Figure 9–26. If your SSIS package needs to support transactions at the task or package level, you configure the TransactionOption property value to either Required or Supported for the task or the package.

■ **Note** You can also configure the `TransactionOption` property to the value `NotSupported` to specifically exclude a component from participating in a package transaction.

If you want to use native SQL transactions in your SSIS package, all tasks participating in the transaction must be configured to use the same Connection Manager. Then, you must configure the `RetainSameConnection` property of that Connection Manager to the value `True`, and finally use the T-SQL `Begin Transaction` syntax within your package. The T-SQL transaction syntax should include a `Begin Transaction` and a `Commit Transaction` statement at minimum. It will typically also include at least one `Rollback Transaction` statement as well; this code is often encapsulated in a T-SQL stored procedure called from an SSIS Execute SQL Task control flow task inside of the particular SSIS package of interest.

Data Profiling

Data profiling is used to help ensure data quality throughout an enterprise. By profiling your data, you can measure the volume and types of inconsistencies your data contains. Data profiles can contain a variety of measurements, including counts, distinct values, missing values, and possible relationships with other data points. If you are integrating data from multiple, disparate systems, data profiling will probably be an important part of your process.

Creating a Data Profile

To begin, create a new package named `DataProfile.dtsx`, and place a Data Profiling Task onto the Control Flow design area. Next, double-click on the task to open the Data Profiling Task Editor. In the Destination drop-down, select “New File connection”. This will open the File Connection Manager Editor. In the “Usage type” drop-down list, choose “Create file,” and in the “File” drop-down, use `DataProfilerResults.xml` to hold your results. Figure 9–27 displays the completed Data Profiling Task Editor.

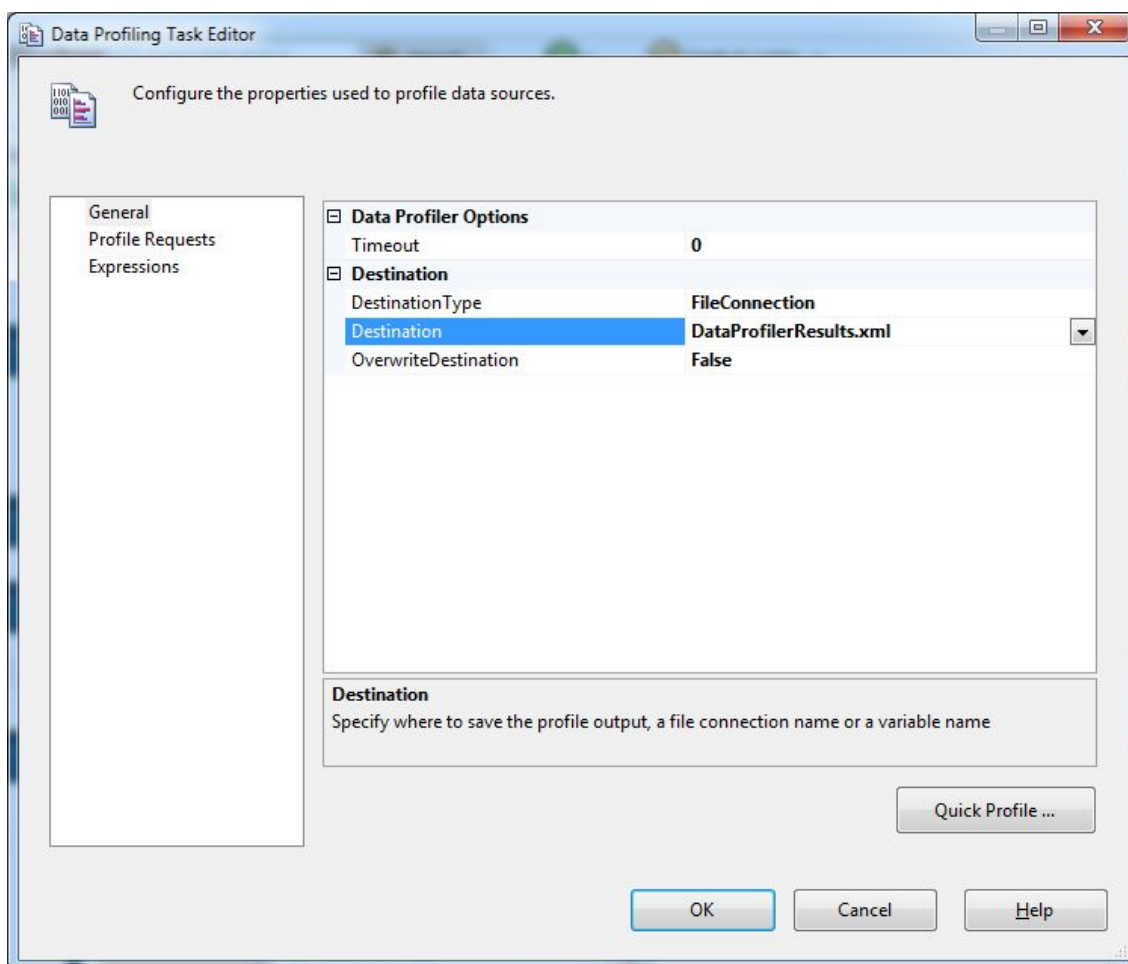


Figure 9-27. The Data Profiling Task Editor, showing completed Destination entries

Now that you have a destination defined for the profile output, click the Quick Profile button to open the Single Table Quick Profile Form. Using this dialog, you will choose a table to profile, and select the types of profiles to generate. The first item you need to complete is to create a new connection, by clicking the New button, and using the Connection Manager dialog to connect to your existing AdventureWorks2008R2 database. Next, select [HumanResources].[Employee] from the “Table or View” drop-down. The “Compute”: check-boxes will have five selections by default, and you will create your profile using these defaults. The profiles you can run via the single table dialog are:

- **Column Null Ratio Profile:** This profile returns the percentage of NULL values in a column.
- **Column Statistics Profile:** This profile returns minimum, maximum, average, and standard deviation for a column.

- *Column Value Distribution Profile*: This profile returns distinct column values, and the percentage of the population that each one represents.
- *Column Length Distribution Profile*: This profile returns the distinct lengths of column values, and the percentage of the population that each one represents.
- *Column Pattern Profile*: This profile uses regular expressions to find invalid, or misformatted, column values.
- *Candidate Key Profile*: This profile finds candidate keys, and any duplicates that exist within the candidate key.
- *Functional Dependency Profile*: This profile finds data dependencies between columns, and any mismatches that exist within the dependent columns.

The Single Table Quick Profile Form is now complete and should resemble Figure 9–28.

Single Table Quick Profile Form

You can profile a table or view on all applicable columns using default settings. Choose the table and the profiles you want.

ADO.NET Connection:

Table or View:

Compute:

- ☒ Column Null Ratio Profile
- ☒ Column Statistics Profile
- ☒ Column Value Distribution Profile
- ☒ Column Length Distribution Profile
- ☐ Column Pattern Profile
- ☒ Candidate Key Profile

for up to

☐ Functional Dependency Profile

for up to

Figure 9–28. The Single Table Quick Profile Form, showing completed profile entries

Click OK accept the profile form and return to the Data Profiling Task Editor, which is now populated with your Profile Type choices. The Profile Requests are shown in Figure 9–29.

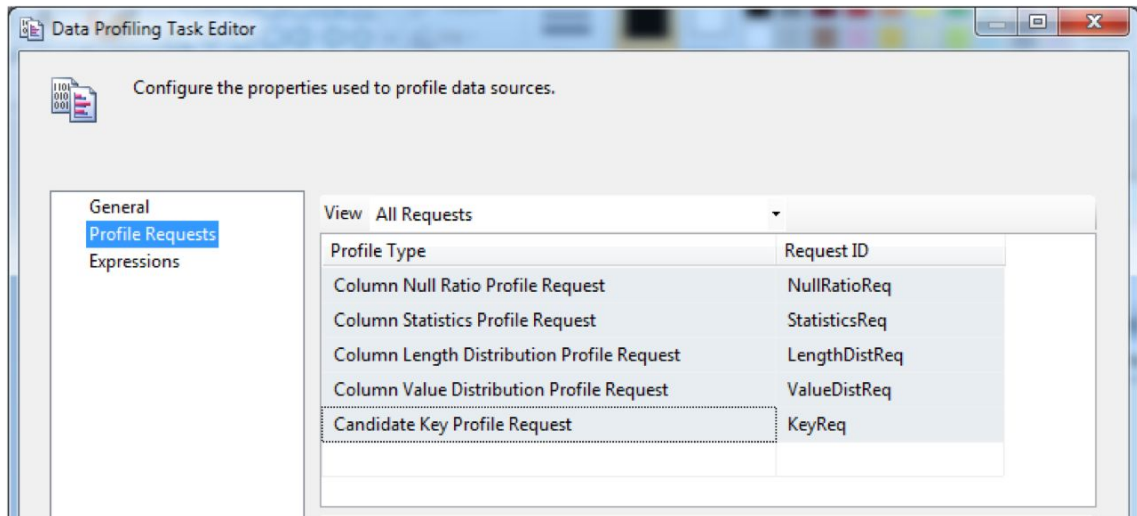


Figure 9–29. The Data Profiling Task Editor, showing the Profile Requests to be executed

Click OK once again, to accept your profile requests and return to BIDS. Your profile of the Employee table is now setup and ready to execute. Go for it!

Viewing a Data Profile

When the package finishes, your data will be profiled and ready for review. To view your profiles in their raw XML format, locate and open the `DataProfilerResults.xml` created by your package in Internet Explorer. A portion of the XML created by the Data Profiling task is shown in Figure 9–30.

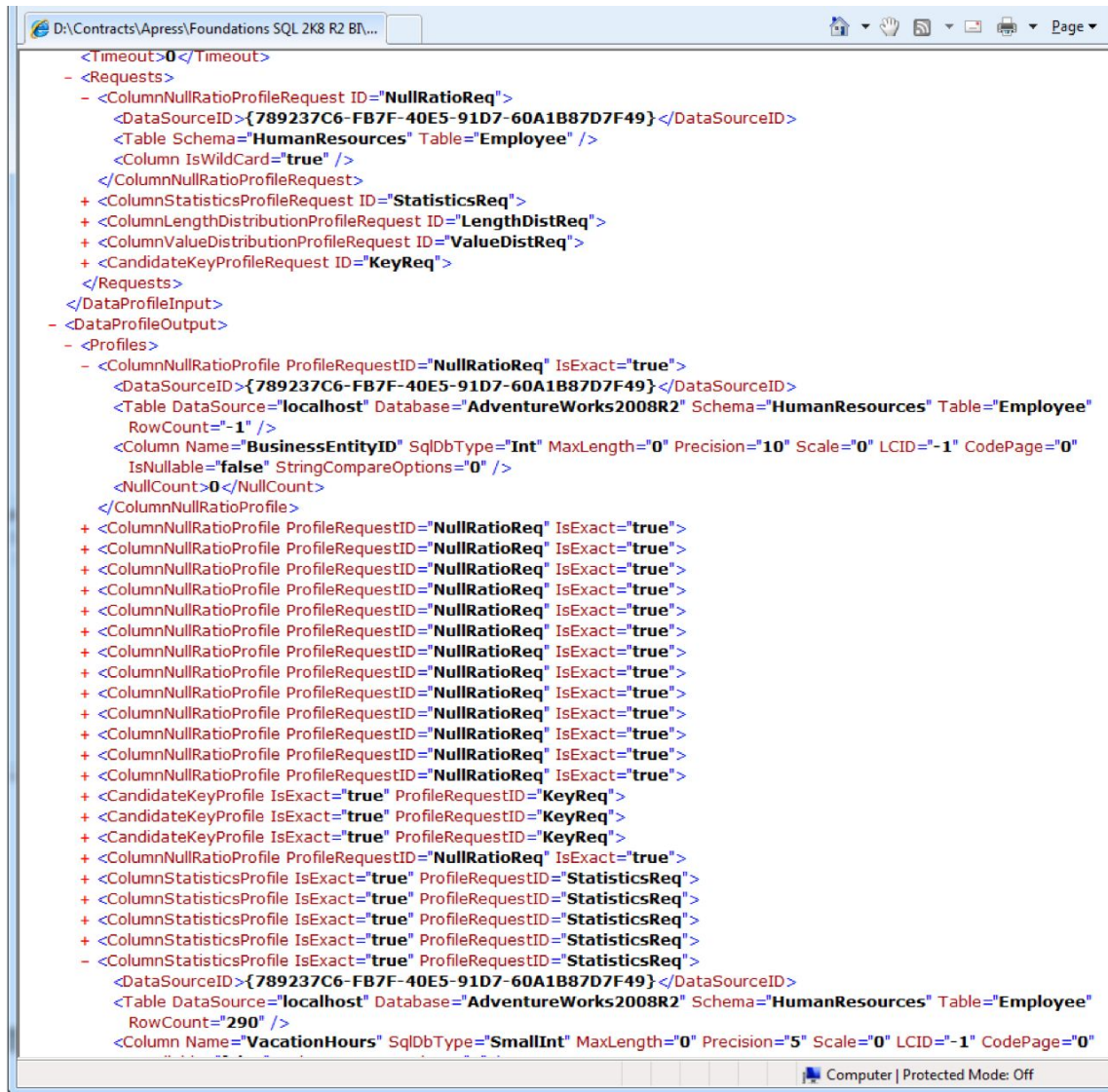


Figure 9-30. A portion of the XML data created by the Data Profiling task

The Data Profile Viewer is a small, stand-alone application that ships with SQL Server to facilitate viewing the XML data created by the profiling task. It is usually located at C:\Program Files\Microsoft SQL Server\100\DTs\Binn\DataProfileViewer.exe. The first steps to viewing a data profile are to launch the viewer and open the XML data you want to review. Next, use the tree structure in the Profiles (Table View) pane to navigate to Tables ► [HumanResources].[Employee] ► Candidate Key Profiles. Finally, in the Candidate Key Profiles pane, select BirthDate under Key Columns. Figure 9-31 shows the Data Profile Viewer with BirthDate selected, and the Key Violations pane populated with duplicates.

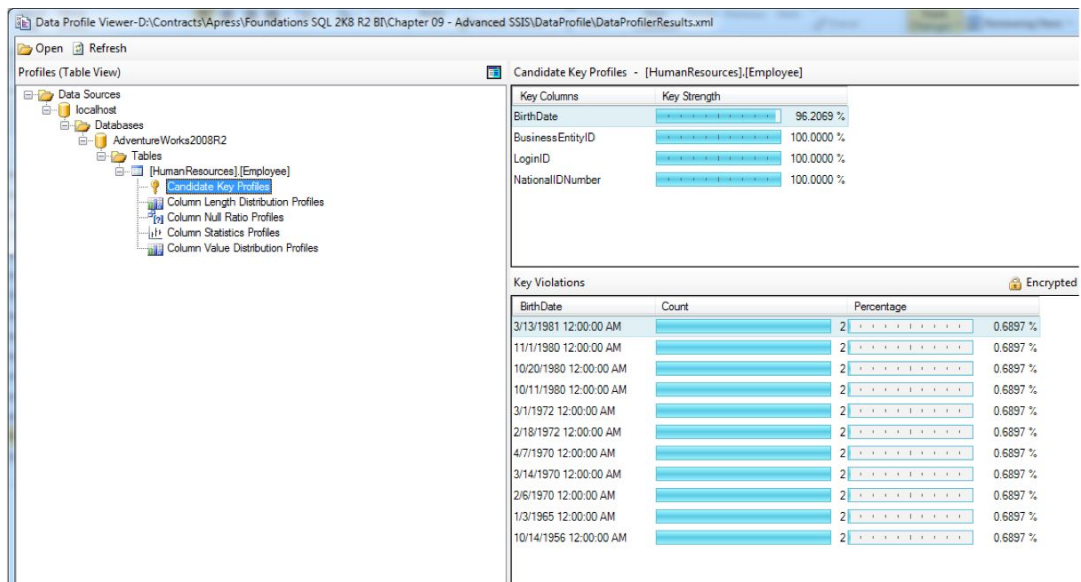


Figure 9–31. The Data Profile Viewer, with the BirthDate candidate key displayed.

Figure 9–31 shows that, while LoginID is unique in our Employee data with a Key Strength of 100%, BirthDate may not be the best choice based on the displayed Key Violations. Another profile, Column Value Distribution Profiles, with Gender selected, is shown in Figure 9–32.

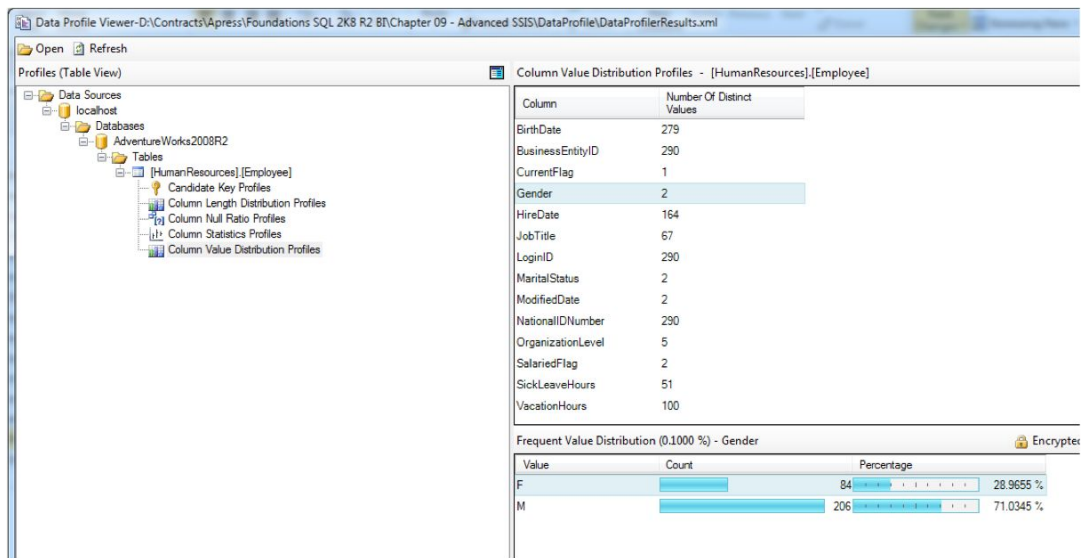


Figure 9–32. The Data Profile Viewer, with distinct column values by gender displayed.

In Figure 9–32, you can see that there are two distinct values for Gender, and 71% of employees in the Employee table have a value of M.

Summary

We've now completed our tour of SSIS. From simple tasks, like using the Import/Export Wizard to quickly move data between a source and a destination, to complex transactional data workflows, SSIS really does it all. SSIS is an invaluable tool for implementing BI solutions.

In the next chapter, we will begin our exploration of reporting tools. Our first stops will be SQL Server Reporting Services (SSRS) and Report Builder 3.0.



Reporting Tools

This chapter will cover the ins and outs of selecting, designing, and implementing one or more reporting clients for your BI solution. The focus of the chapter will be on using Microsoft clients. The selection of an appropriate set of client tools can make or break your BI solution. Ideally, you will select your client strategy near the beginning of your project, as support (or lack of) for various SSAS features in the selected client tools is an important design consideration. Given that backdrop, this is what we'll discuss in the chapter:

- Using Excel pivot tables
- Using SQL Server Reporting Services (SSRS)
- Building your first SSRS report
- Producing reports with Report Builder

In addition, you have the option to work with Office 2010. Because of the amount of functionality included in Office, Chapter 11 is devoted to BI integration in Office 2010 applications.

Using Excel Pivot Tables and Pivot Charts

The simplest choice for reporting is to use an Excel pivot table or pivot chart. These tools offer a good deal of flexibility for creating tabular reports and chart graphics from SSAS. Pivot tables make an easier task of analyzing and filtering your company's data, while pivot charts can create professional visualizations of your prepared pivot tables.

Creating a Pivot Table

A pivot table simplifies the work necessary to create a summarized view of your data. Pivot tables do not require formulas and can make relationships in your data more apparent. Pivot tables can be used to ask questions such as "Who are our best customers?" or "What are our biggest selling products?" To create a pivot table, you need three things: a row field, a column field, and a data field. A row field will summarize data for each distinct row item, for example, customers. A column field will control the width of your pivot table, for example, months or years. Finally, the data field, which will appear in the body of the pivot table, is the data point that your pivot table will summarize.

In this section, you will create a pivot table using the Adventure Works cube. When completed, your pivot table will display customer counts, by country, across time. To create this pivot table in Excel, follow these steps:

1. Open Excel.
2. Choose Data ► From Other Sources ► From Analysis Services.
3. In the Data Connection Wizard, enter your SSAS “Server name”, and click Next, as shown in Figure 10–1.

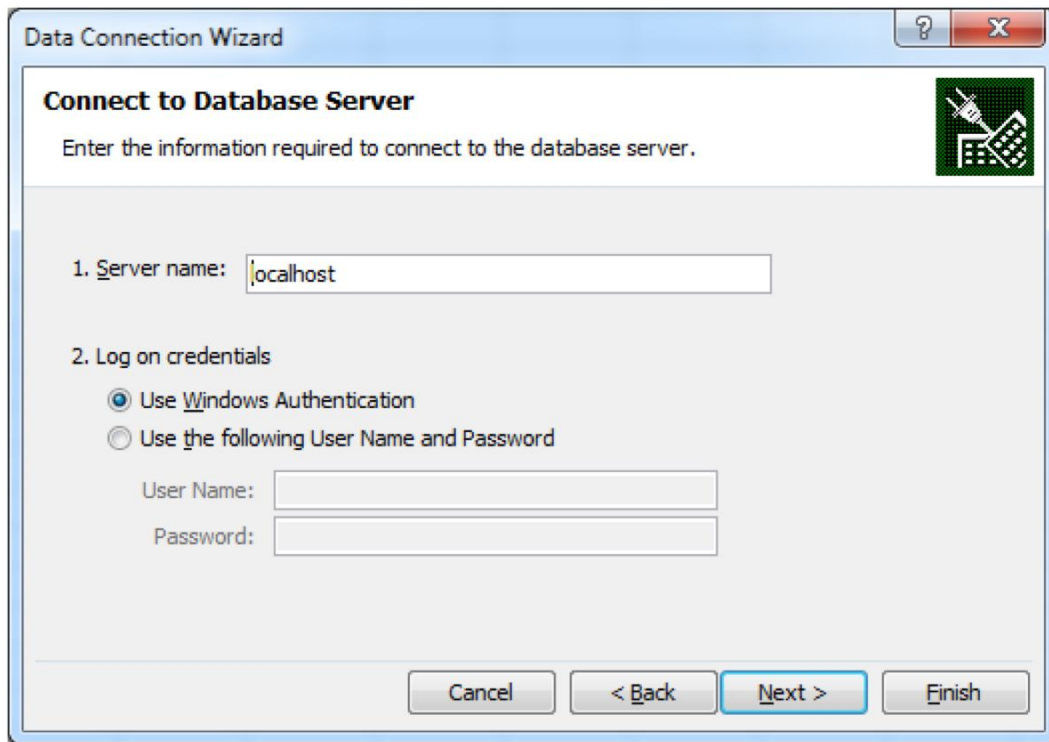


Figure 10–1. Connecting to an SSAS cube in Excel using the Data Connection Wizard

4. Next, select the Adventure Works DW 2008R2 SE database, choose Adventure Works cube, and click Next, as shown in Figure 10–2.

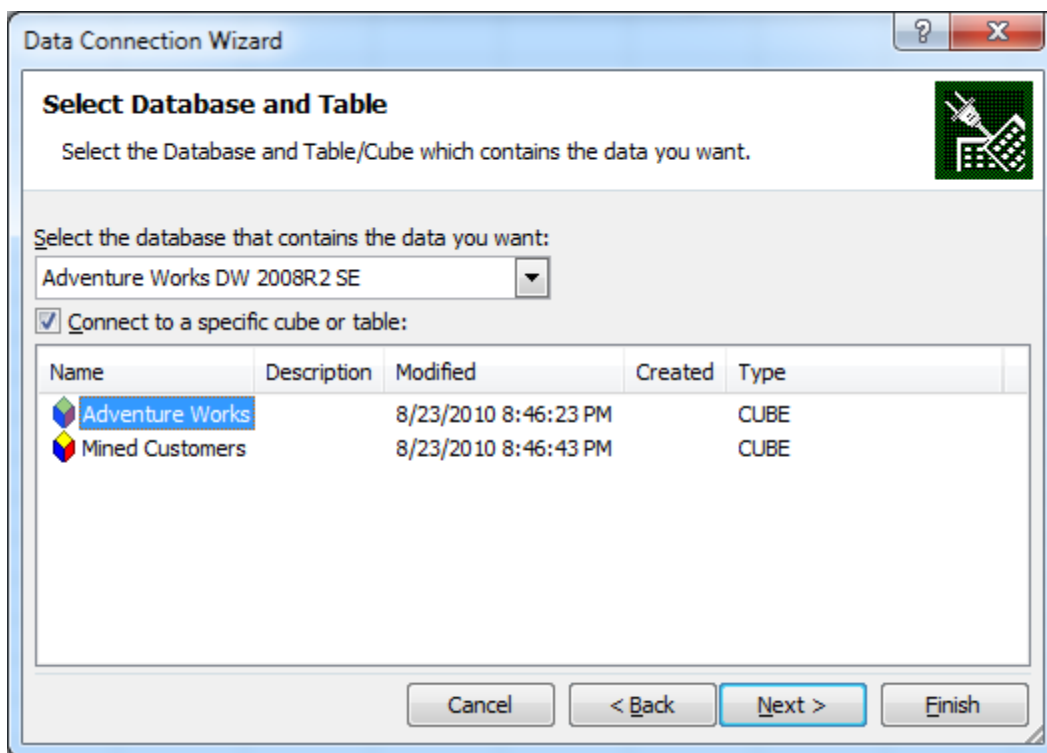


Figure 10–2. Choosing a database and cube using the Data Connection Wizard

5. Review the final wizard dialog, and click Finish.
6. In the Import Data dialog, select PivotTable Report as your data view, and place your data into the existing worksheet with the formula =Sheet1!\$A\$1. Click OK.
7. From the PivotTable Field List, drag the items that you want to show in your PivotTable to the Drag Field area. Figure 10–3 shows an example using Calendar Year as the column label, Country as the row label, and Customer Count for the values.

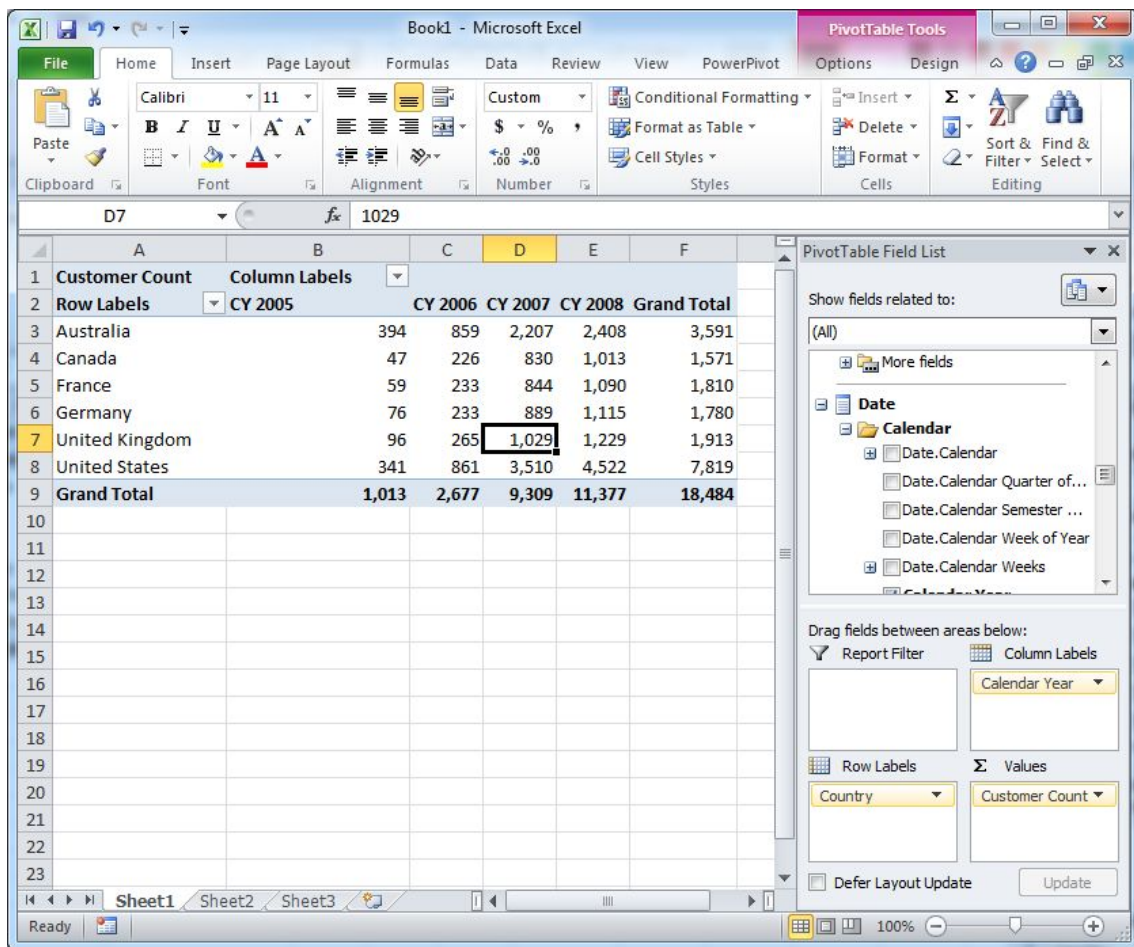


Figure 10-3. A PivotTable using Calendar Year, Country, and Customer Count

Creating a Pivot Chart

A pivot chart enhances Excel's pivot table functionality, giving you the tools to create charts directly from your pivot tables. Excel contains a variety of chart types, including the standard column, bar, line, and pie.

In this section, you will add a pivot chart to the worksheet you finished in the last section. To add a pivot chart to this worksheet, follow these steps:

1. Choose Options ► PivotChart from the PivotTable Tools menu area.
2. Choose Stacked Column in 3D (top row, fifth from left) from the Column pane.
3. Click OK. Figure 10-4 displays the current worksheet, with your newly created pivot chart added.

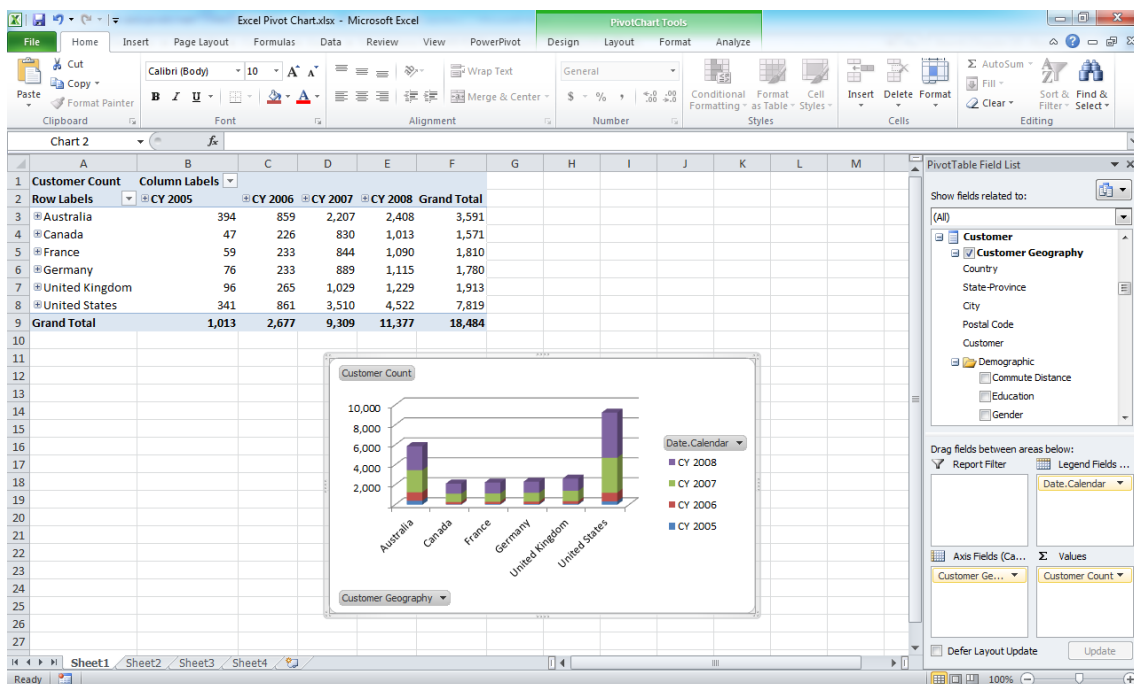


Figure 10–4. Your pivot chart, created in the same worksheet as your pivot table

Publishing Your Workbook

You also have the option to save your pivot table or chart as a web page. This allows your end users to review the resulting pivot chart using a standard web browser. To publish your pivot chart, choose **File** ➤ **Save As**. In the **Save As** dialog box, choose **Single File Web Page** from the “Save as type” drop-down list. Name your file **CustomerCountByCountry.mht**, and click **Save**. Figure 10–5 shows your worksheet in Internet Explorer.

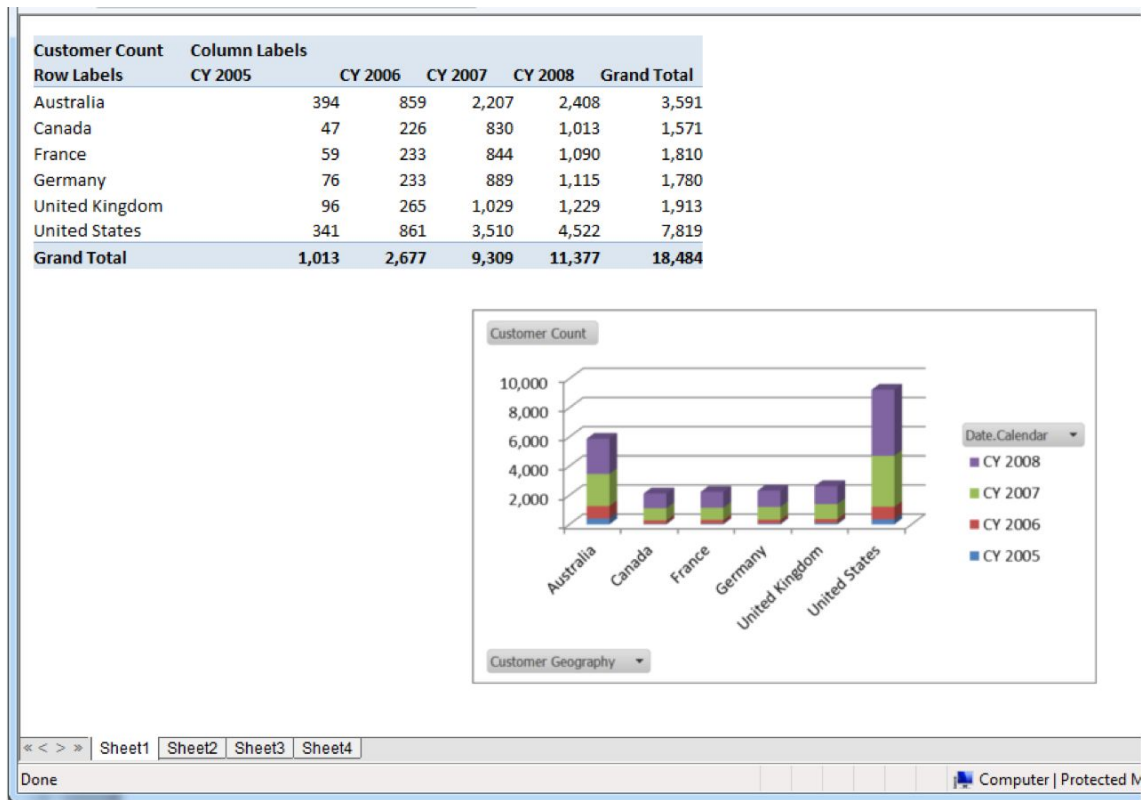


Figure 10–5. The Customer Count By Country workbook displayed in Internet Explorer

Using SQL Server Reporting Services

SQL Server Reporting Services (SSRS) is an enterprise-capable reporting solution for all types of data sources, including relational, multidimensional, XML and text. This section mostly focuses on using SSRS as a reporting client for SSAS cube data. However, before we do that, let's discuss SSRS in general.

SSRS Components

SSRS consists of the following default components:

- *Report Manager*: This is the access and management tool that provides web-based report functionality and the default user interface for the results of the web service. End users access an ASP.NET web application, which is also called Report Manager, by navigating to the (default) URL:
<http://localhost/Reports/Pages/Folder.aspx>.

- *Report Server Web Service*: This required component is one of the areas where the core real-time, on-demand report and model processing is done within SSRS.
- *Scheduling and Delivery Processor*: This component handles background scheduling tasks, such as scheduled report generation and delivery.

■ **Note** You can develop alternate user interfaces, such as web and Windows forms, by connecting to published reports hosted on the report server or by writing your own .NET applications, which call the report server web service APIs.

- *Report Designer*: This is a design tool that runs in BIDS, which is a shell integrated with Visual Studio, or as a separate application. BIDS is the default development environment for report queries, layout, and structures. You can use other report development tools to author reports, including Report Builder, discussed later in this section.

SSRS Reporting Samples

Several SSRS reports are included with the SQL Server samples. These are located, by default, at C:\Program Files\Microsoft SQL Server\100\Samples\Reporting Services\Report Samples\AdventureWorks Sample Reports. The included samples contain not only sample reports but also interesting samples of other functionality, such as SSRS management and custom delivery and authentication. The default location for all SSRS sample code is C:\Program Files\Microsoft SQL Server\100\Samples\Reporting Services.

Building Your First SSRS Report

In this section, you will learn the steps involved in building a report by building one from scratch with the sample AdventureWorks cube as a data source. To build a report against an SSAS cube for SSRS, you open BIDS and choose File ► New ► Project. In the New Project dialog box, you'll select the Business Intelligence Projects project type. You have three template types to choose from when building reports: Report Server Project Wizard, Report Server Project, or Report Model Project. For this discussion, we'll use the Report Server Project Wizard template, as shown in Figure 10–6.

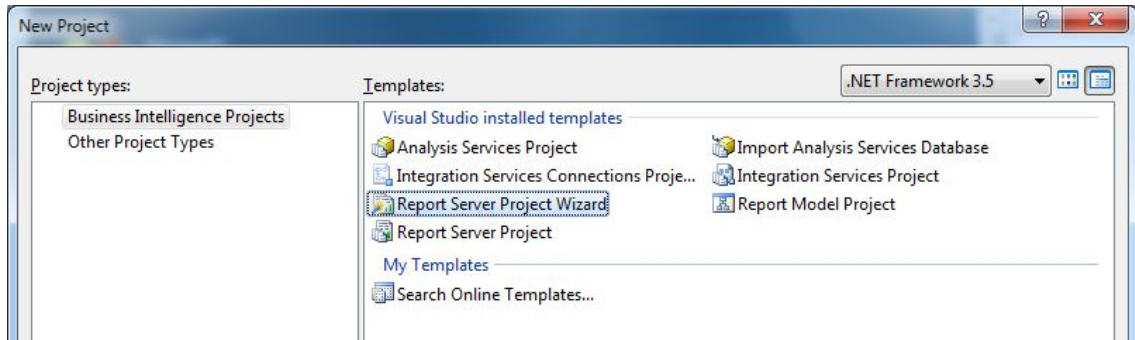


Figure 10–6. To create a report, you open BIDS and select one of the three types of report templates.

After you select this template type, enter a project name, location, and solution name. Click OK. The wizard presents you with a series of dialog boxes.

Running the Report Server Project Wizard

After the welcome screen, the Select the Data Source dialog will appear; there, you are asked to configure a data source. Although this process is similar to configuring a data source in SSAS, there are some important differences. One of these differences is the option to create the data sources as shared, or global to the entire report solution. It is a best practice to create shared data sources, because you'll spend less time creating connections (that is, you can just reuse the global one you've created for multiple reports), and you can easily update any of the values (that is, the connection string information) in the data source, which will affect every report in the project.

You can either type the desired connection string directly into the Connection String list box, or you can use the Edit or Credentials buttons to quickly generate a connection string based on your input values. Figure 10–7 shows the completed data source dialog.

■ **Note** If you click the Credentials button, the only available connection type is Use Windows Authentication (Integrated Security). The other choices—"Use a specific username", "Prompt for credentials", and so on—are grayed out and not available for connections to SSAS using SSRS.

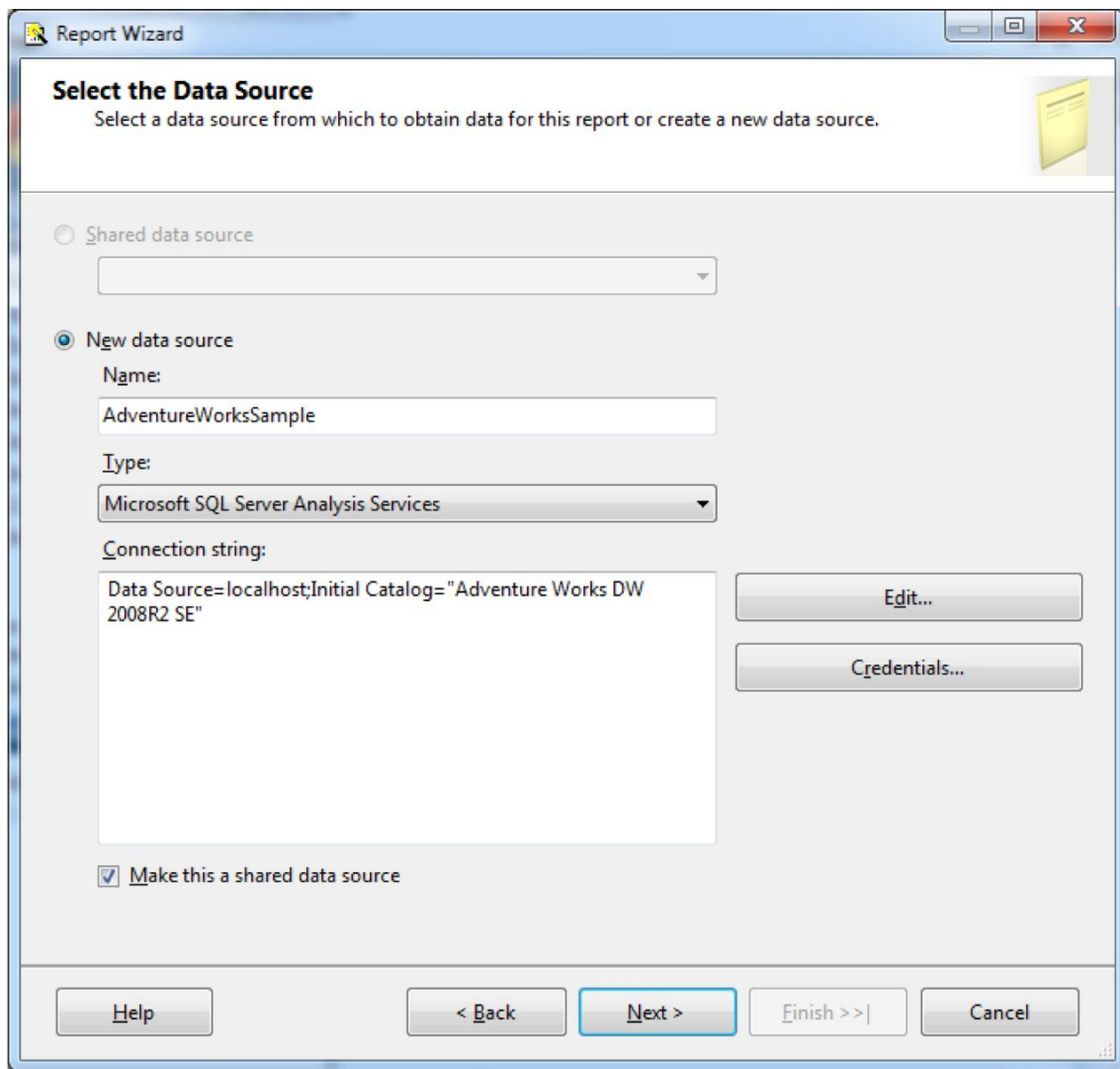


Figure 10–7. In the *Select the Data Source* dialog box, you configure the connection to SSAS.

Click **Next** to advance to the *Design the Query* dialog. The next step is to write the MDX query to retrieve the values from your cube.

Designing the Query

The Report Wizard has a built-in visual query designer, which you access by clicking the **Query Builder** button. Query Designer, shown in Figure 10–8, is much like the cube browser in BIDS. To generate an

MDX query, you simply drag and drop items from the Metadata panel onto the design areas. This built-in visual MDX query designer greatly reduces the amount of manual query writing you'll need to do to build a reporting solution for SSAS, which gives you the ability to generate reports much more quickly.

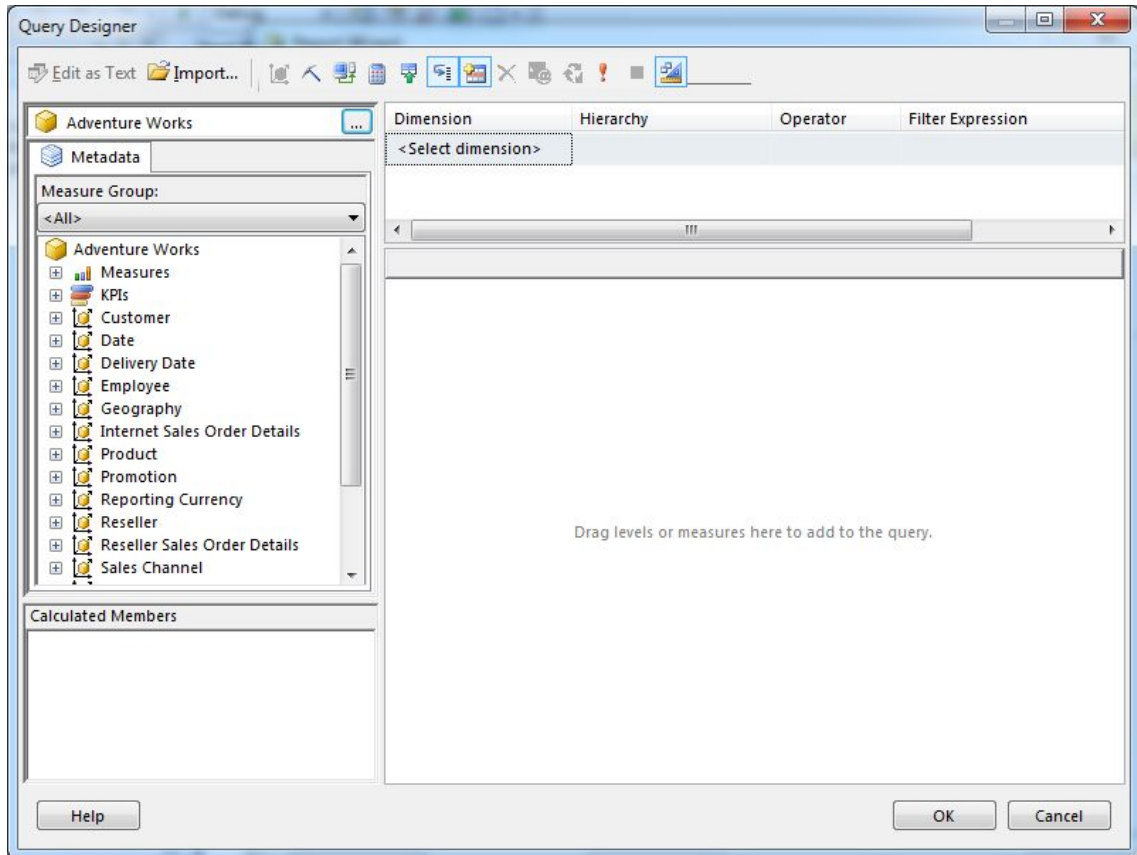


Figure 10–8. The SSRS Report Wizard for SSAS includes a built-in visual MDX query builder that greatly reduces the need for you to manually write MDX queries.

The visual query designer has two modes of operation: design and native query. It opens in design mode. This means that you drag and drop items from the Metadata viewer to the blank windows on the right to generate a MDX query. If you want to manually write (or use) an MDX query, click the Design Mode button (the last button on the toolbar) to switch to native query mode. Then, you can type MDX natively in the Query Designer interface in combination with dragging and dropping items from the Metadata browser.

Figure 10–9 shows a drag-and-drop query. Note that you can enable parameters in your report interface by simply checking the box in the filter section at the top of the query design work area. These parameters can be presented in the user interface as a blank text box or a drop-down list (showing a list you provide or one that is generated based on another query) that shows a default value. Multiple query

parameter values can be selected as defaults and can be passed by the end user by making multiple selections in the drop-down list.

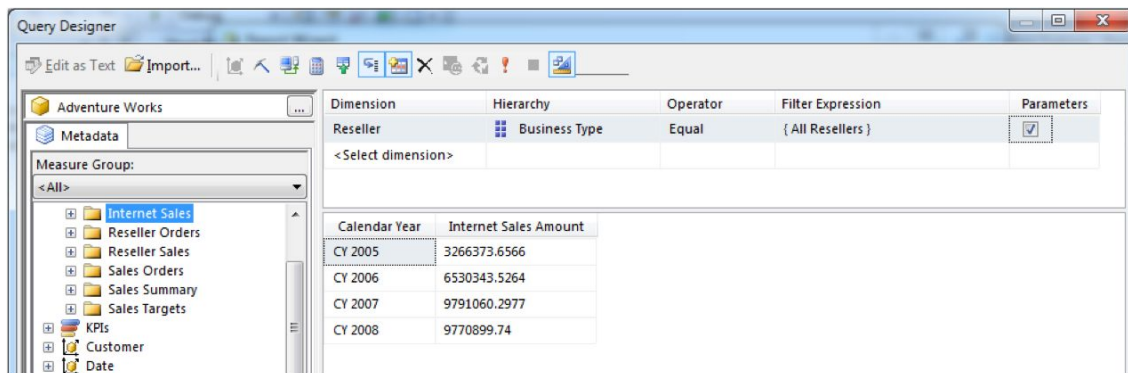


Figure 10–9. The simplest way to create a MDX query in Query Designer is to drag and drop items from the Metadata viewer to the design surface. Note that you can enable parameters by simply checking the box under the Parameters column at the top of the design surface.

You can also create and add MDX-calculated members while in visual design mode by right-clicking the Calculated Members area at the lower-left of the design surface and selecting New Calculated Member. The Calculated Member Builder dialog box appears, so you can write the MDX for the calculated member (see Figure 10–10). Normally, you will generate, rather than manually author, the MDX.

Remember that the calculated members you are creating here are specific to the particular report that you are creating. In other words, they are visible only to that report. You can think of this type of calculated member as local. This differs from creating calculated members as objects for a particular cube using the cube designer (Calculations tab) in BIDS; calculated members created in the cube can be considered global.

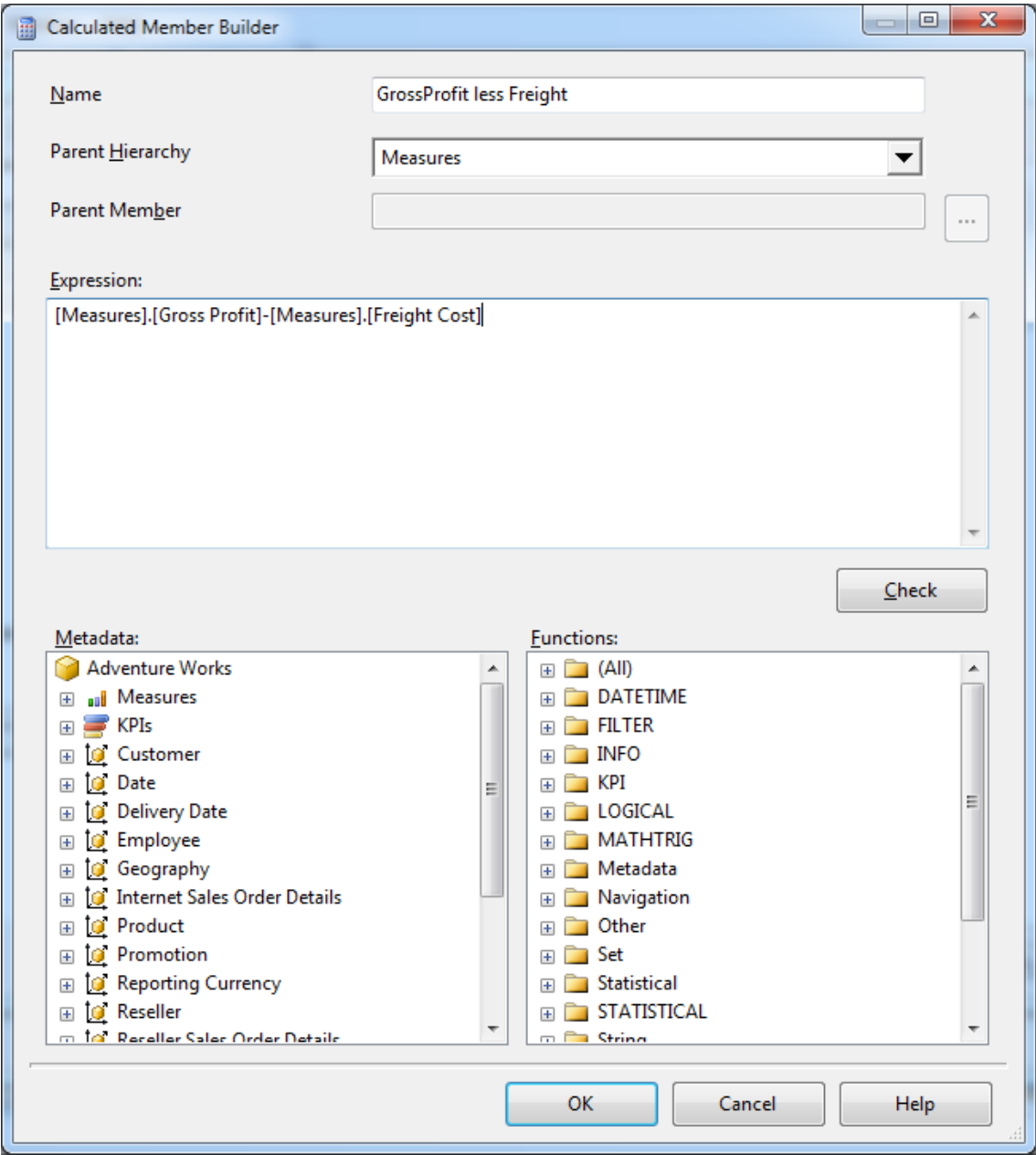


Figure 10–10. You can add calculated members to your report by using the Calculated Member Builder.

Tip To generate, rather than manually write, the MDX calculated member expression, you can drag and drop measures, dimensions, and so on from the Metadata area of the Calculated Member Builder dialog box to the Expression area. You can also drag or double-click Functions to add them to the expression that you are building.

Figure 10–11 shows the same query as was generated in Figure 10–9, now rendered in the Query Builder so that the MDX statement is visible. You can edit this query by simply typing directly into the interface. Note also that the Metadata viewer includes the MDX Functions and Templates libraries. As in BIDS and SSMS, you can also generate the MDX query by selecting and then dragging and dropping Metadata items, Functions, or Templates into the MDX work area.

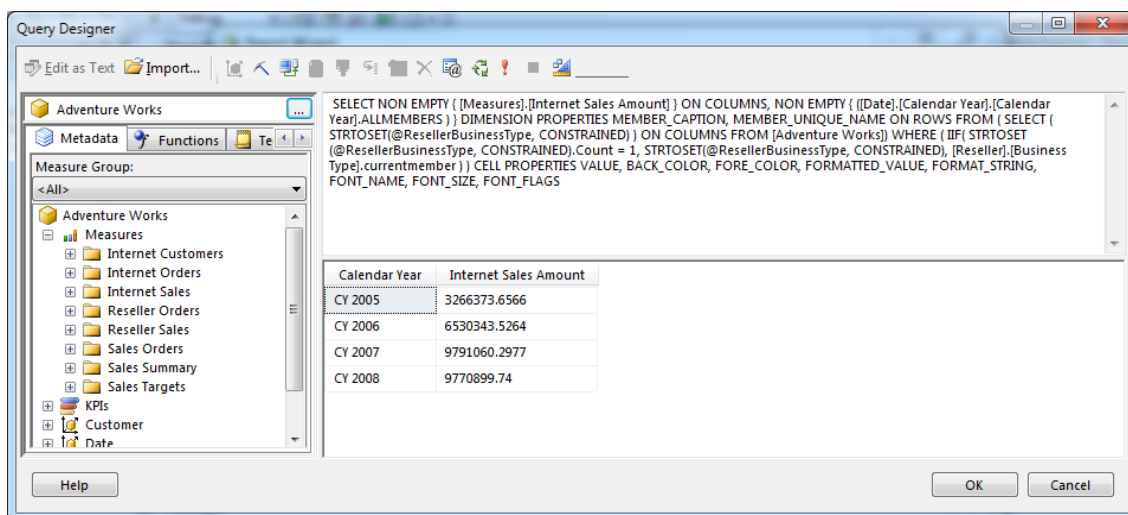


Figure 10–11. You can switch from visual design to native MDX query mode in the Query Designer by clicking the Design Mode button (the last button) on the toolbar.

When you are working in native query mode, two additional buttons become available on the toolbar: Query Parameters (fifth button from the right) and Prepare Query (fourth button from the right). When you click the Query Parameters button, the Query Parameters dialog box, allowing you to visually configure the query parameters, becomes available (see Figure 10–12). Here, you can specify parameter names; associate those names to dimensions, attributes, or hierarchies (using the Hierarchy section); allow multiple values; and set a default value.

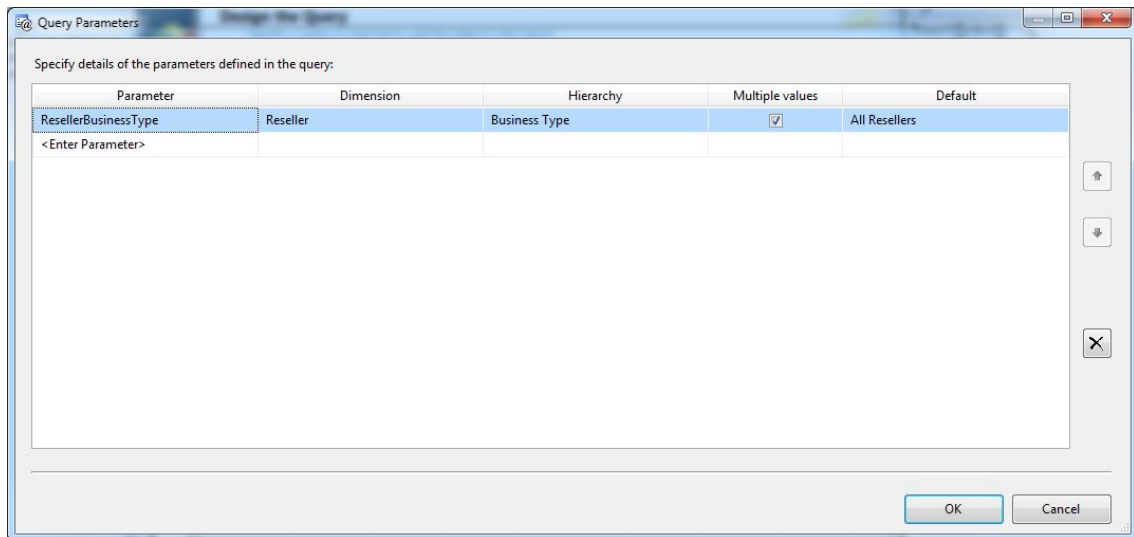


Figure 10–12. When working in native MDX query mode in the Query Builder, you can use the Query Parameters dialog box to view and adjust the query parameters as needed.

The Prepare Query button acts much like the parse (blue checkmark) button in SSMS T-SQL query mode; that is, when you click it, the query syntax is checked, and any errors are returned to you via a pop-up window. The one item that is lacking in this view is also similar to a missing feature in the graphical and text-based (SQL) query designers—IntelliSense.

The next few dialog boxes of the wizard ask you a series of questions about the visual design of the report. This process is very similar to other visual report design wizards, particularly Access's. For this report, select Tabular from the Select the Report Type dialog, and click Next. From the Design the Table dialog, simply select Next. The Choose the Table Style dialog will have the Slate theme selected by default; use this theme by clicking Next. Finally, click Next in the Choose the Deployment Location dialog, name your report “Internet Sales per Year”, and click Finish to complete the Report Wizard.

Previewing and Designing Your Report

After you’ve run through the wizard, you will have two work areas to continue to work with your report: the Preview and Design tabs. The Preview tab for the “Internet Sales per Year” report is shown in Figure 10–13.

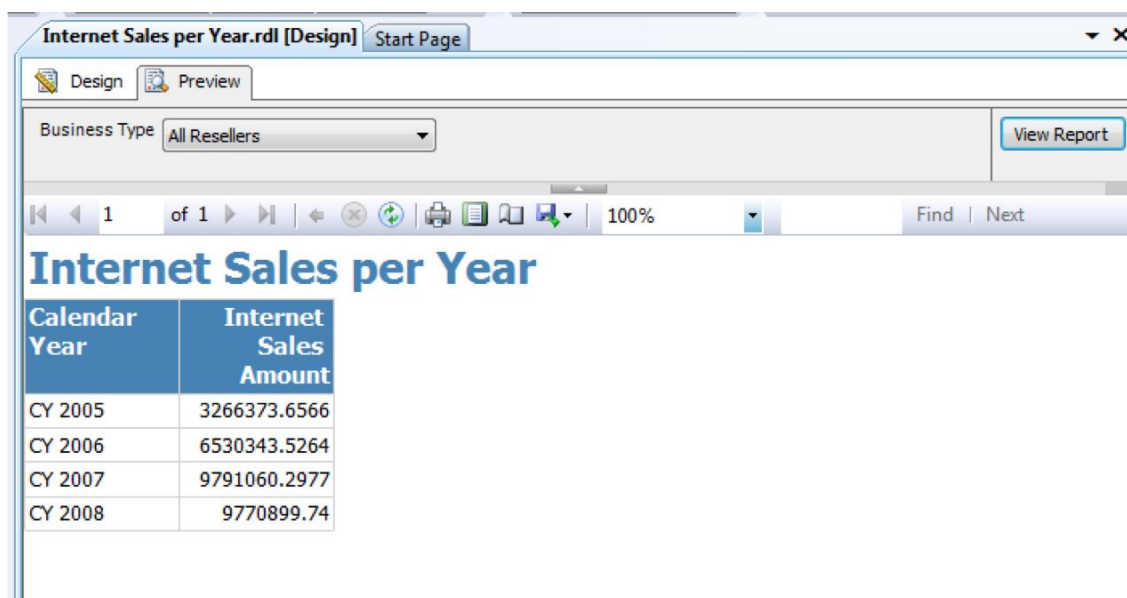


Figure 10–13. The Report Design interface includes a Preview tab so that you can review the report you've created as you continue to enhance the design.

If you want to alter your query, first select the Design tab, and ensure that the Report Data pane is visible. The Report Data pane can be opened by clicking View and selecting Report Data. Next, right-click the Dataset you want to edit in the Report Data pane, and select Query to load the Query Designer, where you can make changes to the MDX query either by dragging and dropping items from the Metadata view (using visual design mode) or by a combination of typing MDX and dragging items from the Metadata, Functions, or Templates tabs (using native MDX query mode). Of course, if the changes you make to the MDX query invalidate the report layout, for example, if you remove an item from the On Columns clause of the MDX query, you would also have to update your report by using the Layout tab to remove that information from the report layout.

You can associate more than one query with a report. To do this, you right-click the Datasets node in the Report Data pane, and select Add Dataset. Associating more than one query with a report is commonly done to populate parameter lists with dynamic values.

After you're satisfied with the values retrieved by your MDX query (or queries), you may want to modify the layout of your report. To do so, click the Design tab in the report designer. While there, you may place the available data fields onto the available design surfaces. The example uses a simple table. Your choices are Table, Matrix, Chart, List, or Subreport. We added a total amount by right-clicking the Internet Sales Amount data cell and choosing Add Total. Next, type TOTAL in the Calendar Year cell to the left of your Internet Sales Amount total. Note that the `sum()` function was automatically applied to the field value, as shown in Figure 10–14.

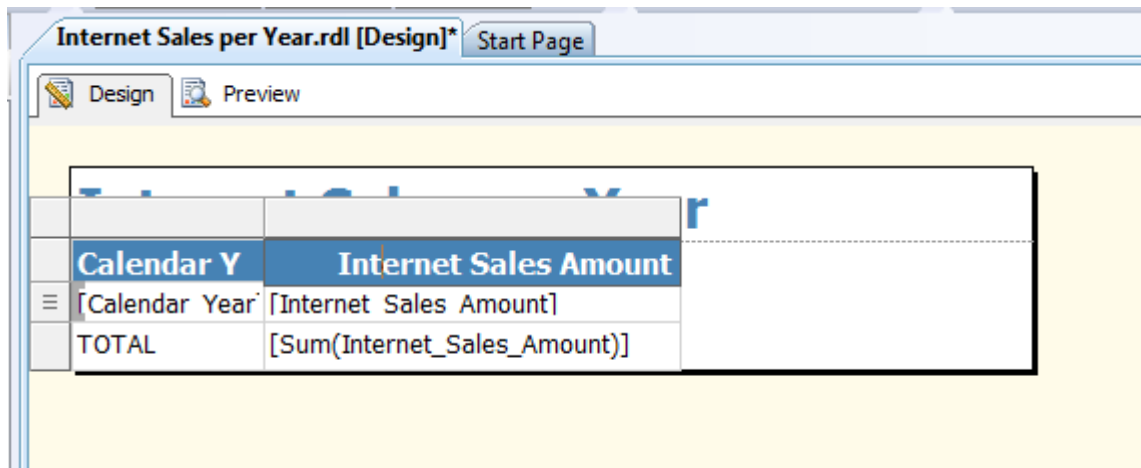


Figure 10–14. *The Internet Sales per Year report, as seen on the Design tab*

Another area you may want to work in while designing your report is the Report Parameters Properties dialog. To work with parameter properties, right-click a parameter in the Parameters node of the Report Data pane, and select Parameter Properties. You can then further configure the parameters per the values available in the dialog box; for example, you can allow null values, blank values, multiselects; or populate a drop-down list with values from the cube (see Figure 10–15).

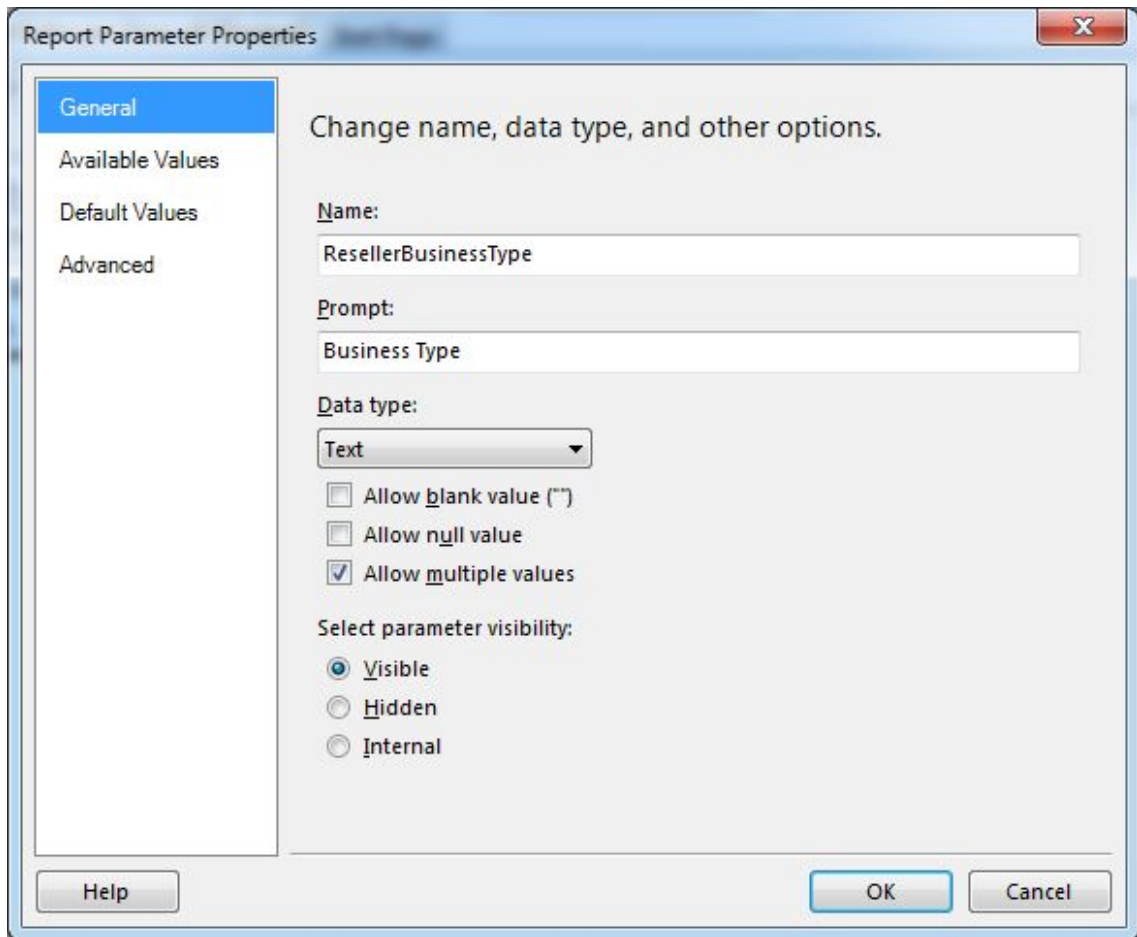


Figure 10–15. The Report Parameter Properties dialog box allows you to configure a parameter for your report.

Publishing Your Report

When you're satisfied with your report, you can now deploy it to the Report Server web service. To do this, first right-click the name of your solution in the Solution Explorer window, and select Properties to open the Property Pages dialog for your solution. Next, verify that the TargetServerURL address is correct for your particular installation of SSRS. The example shown in Figure 10–16 will deploy your project to `http://localhost/ReportServer`.

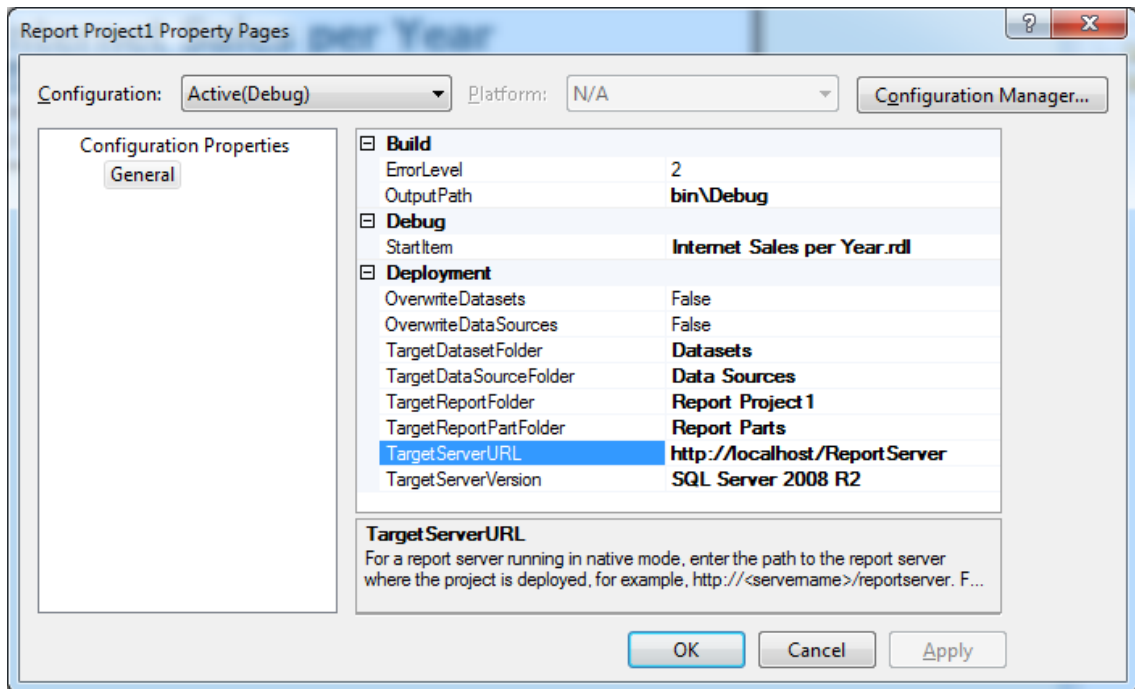


Figure 10–16. Be sure to verify the *TargetServerURL* location on the Report Project Property Pages prior to attempting to deploy your reporting project.

The *StartItem* property lists the first report that you created in the project. Note that the file name ends in *.rdl*. This is the XML dialect Report Definition Language (RDL), which the Report Designer generates in Visual Studio. When you build or deploy a report or report project, the *.rdl* syntax is validated against a specialized type of XML Schema document (XSD). Be aware that *.rdl* and *.rds* files are not compiled during this process.

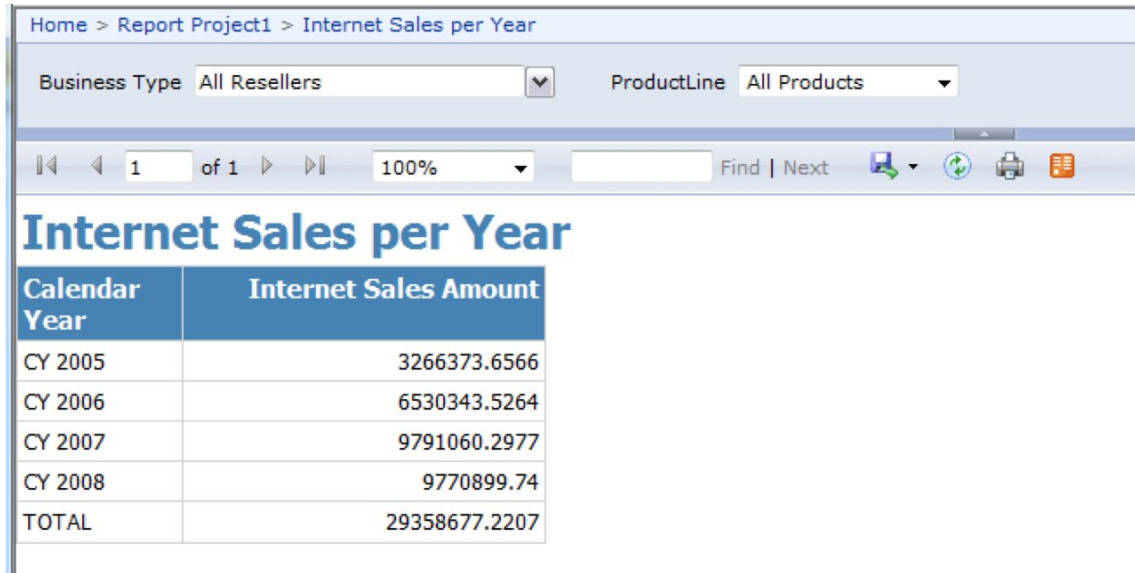
You can actually see, and manually edit, the RDL being generated in Visual Studio by right-clicking any report in Solution Explorer and choosing View Code. You will want to avoid editing the *.rdl* files manually whenever possible, because it is easy to make a mistake that will invalidate the entire file. Be sure to back up an *.rdl* file *before* editing it manually. Remember that RDL, being an XML dialect, is both case sensitive and more sensitive to whitespace than some other formats. A portion of the RDL for a report is shown in Figure 10–17.



Figure 10–17. To view or edit the RDL associated with a report, right-click the report, and click View Code.

If you have made errors in configuring your report, those errors may appear when you attempt to deploy the project. Any design-time errors will be shown in BIDS at the bottom on your work area in the Error List window. These errors result when you create a configuration that is invalid for the schema, for example, binding a field in a table using the Layout tab with an expression that contained invalid syntax. You can also create report configurations that result in runtime errors (these are most often due to incorrect data source connection string information). You will discover these types of errors when you attempt to view the report you've created in BIDS by using the Preview tab. BIDS displays a description of the runtime error on the Preview design surface.

After you've successfully deployed your report, you can then view it using the Report Manager Web interface installed with SSRS, or you can use the web service APIs to create your own customized user interface if the included Report Manager Web site doesn't meet your business requirements. Your end users can also use the available drop-down list on that web site to render the report in formats other than the default HTML. These formats include .xls, .pdf, .csv, and .xml. Also, you can configure security and other runtime settings (for example, execution and caching behavior) via Report Manager. Figure 10–18 shows a sample report rendered in Report Manager.



| Calendar Year | Internet Sales Amount |
|---------------|-----------------------|
| CY 2005 | 3266373.6566 |
| CY 2006 | 6530343.5264 |
| CY 2007 | 9791060.2977 |
| CY 2008 | 9770899.74 |
| TOTAL | 29358677.2207 |

Figure 10–18. You can use Report Manager to present your SSAS reports to end users.

Another method of producing reports from SSAS cube data sources using SSRS is to use the new Report Builder interface—more about that in the next section.

Producing Reports with Report Builder

The Report Builder tool included with SSRS is a Windows application that you, or authorized users, can launch directly from Report Manager. Report Builder provides you (or authorized end users) with a simplified interface for report development. Using Report Builder is an alternative to creating report definitions using BIDS or Visual Studio.

Creating a Report Model

Before you launch Report Builder, you must first create a report model. A *report model* is a specific type of an XSD schema called Semantic Model Definition Language (SMDL), which is generated for SSAS cubes by clicking a button in Report Manager. An SMDL model is a way to describe the underlying database in business terms, so the end user does not need to know about tables, relationships, measures, dimensions, and the like.

There is only one way to create a report model for SSAS (unlike for models run against SQL Server data, where you have multiple methods of creating a report model)—using the GUI interfaces and tools. You do this by generating the schema from inside Report Manager. First, you must navigate to the data source properties for the SSAS connection in Report Manager. Once there, you click the Generate Model button on the toolbar at the top of the page to create a model that can be used by the Report Builder. This interface is shown in Figure 10–19.

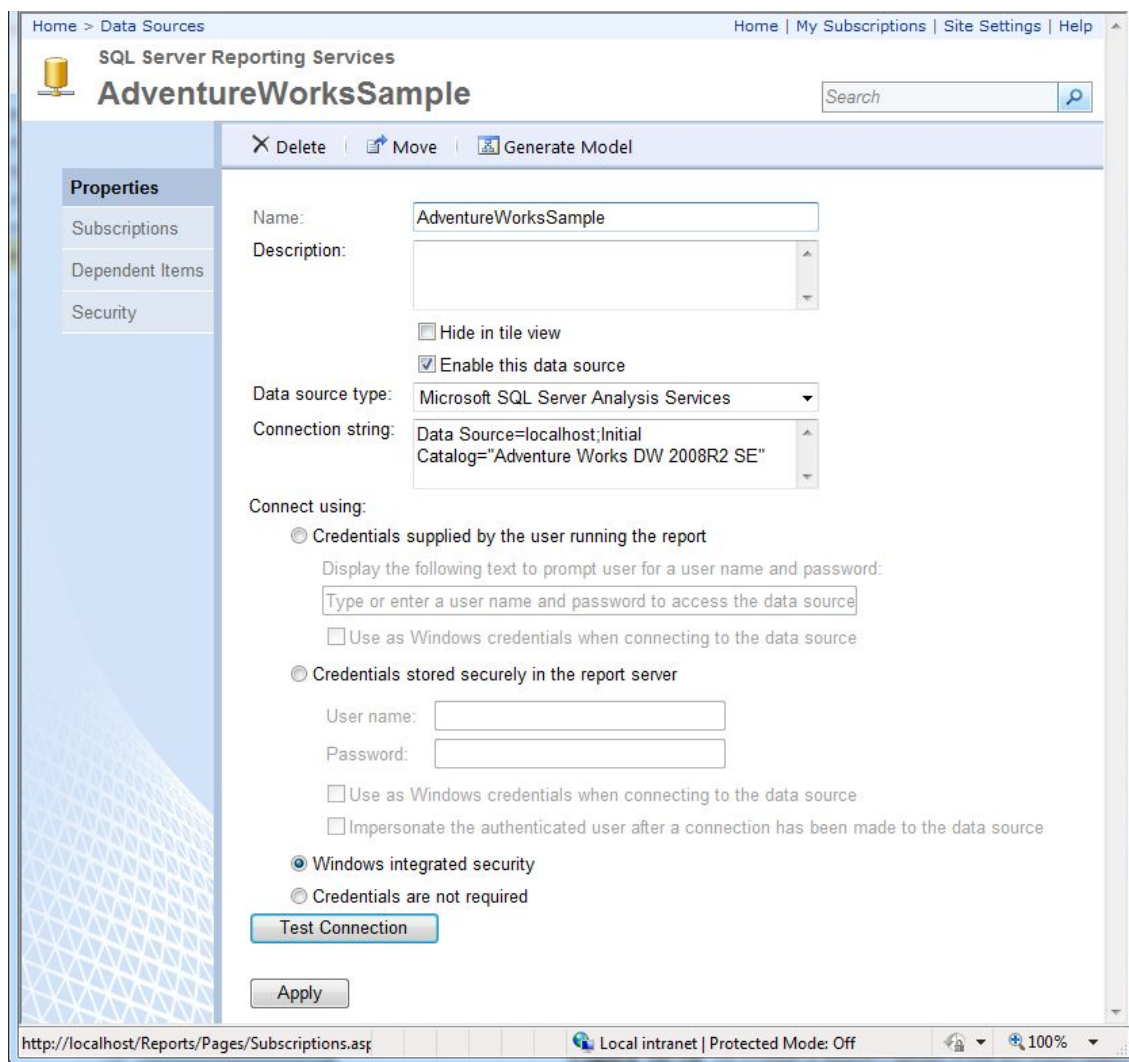


Figure 10–19. To create a Report Model from an SSAS data source, you navigate to the properties of the data source and then click Generate Model.

After you've created a model, you can \ launch the Report Builder application by clicking the Report Builder button on the Report Manager toolbar. The button is associated with this URL by default: <http://localhost/reportserver/reportbuilder/reportbuilder.application>. You could, of course, access this URL by embedding the link in your own custom client application as well. After you click the link and agree to run the application, the application downloads from SSRS and launches. You must have permissions to run this application as well. For more information about the required permissions, see the BOL topic "How to Configure Report Builder Access".

Creating a Dataset

The first step in using the Report Builder is to select the report model source that you want to use as a basis for your report. Begin by selecting New Dataset and choosing "Browse other data sources". Navigate to the Data Sources directory, and open the AdventureWorksSample model you created in the last section. Click Create, followed by OK in the Choose Perspective dialog.

The next choice is which entity you want to use as the basis for your dataset. Report Builder reports are built around the concept of displaying a single entity and some or all of its related attributes (or fields). So, what's an entity? An *entity*, according to Books Online, is "a logical collection of model items, including source fields, roles, folders and expressions, presented in familiar business terms." For SSAS Report Builder models, entities are created for each dimension and for most fact tables. Another way to think of an entity is as a particular business object, such as customer, product, or marketing campaign.

To continue working toward building a report using Report Builder, you select an entity and then select the desired field from the available (associated) fields (or attributes) list, which is shown in the Fields box. To use a particular entity, you need to drag it to the design surface. After you drag an entity to the design surface, then the available entities displayed are restricted to the particular selected entity and all child entities associated with it. If you want to reset the Entities list back to the original (master) set, you simply drag all items off of the report design surface. Figure 10–20 shows the Entities and Fields selection list boxes in the Report Builder.

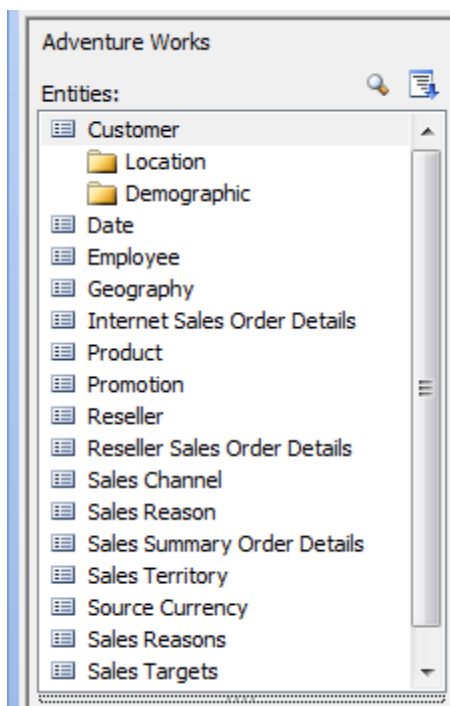


Figure 10–20. To configure your report, you drag an entity from the Entities list to the report design surface. You then drag associated fields (or attributes) to the design surface.

Use the Entities and Fields panes to place Calendar Year and Internet Sales in the drag-and-drop column fields area of the designer. When finished, save your dataset as InternetSales.

Creating a Report

Now that you’ve created a dataset, you can use the InternetSales dataset as the basis for a new report. To begin, click the radial menu, and click New to open the New Report or Dataset dialog. Next, select Table or Matrix Wizard. In the New Table or Matrix dialog, select “Choose an existing dataset”, and browse to the InternetSales dataset you saved in the last section. Ensure that InternetSales is selected, and click Next. In the Arrange Fields dialog, drag and drop Internet_Sales_Amount into the Values area, and Calendar Year into the Row groups area. Click Next. Confirm that the “Show subtotals and grand totals” check box is checked in the “Choose the layout” dialog, and click Next. Finally, select Slate in the Styles pane, and click Finish. Figure 10–21 shows the Report Builder GUI with your new report definition.

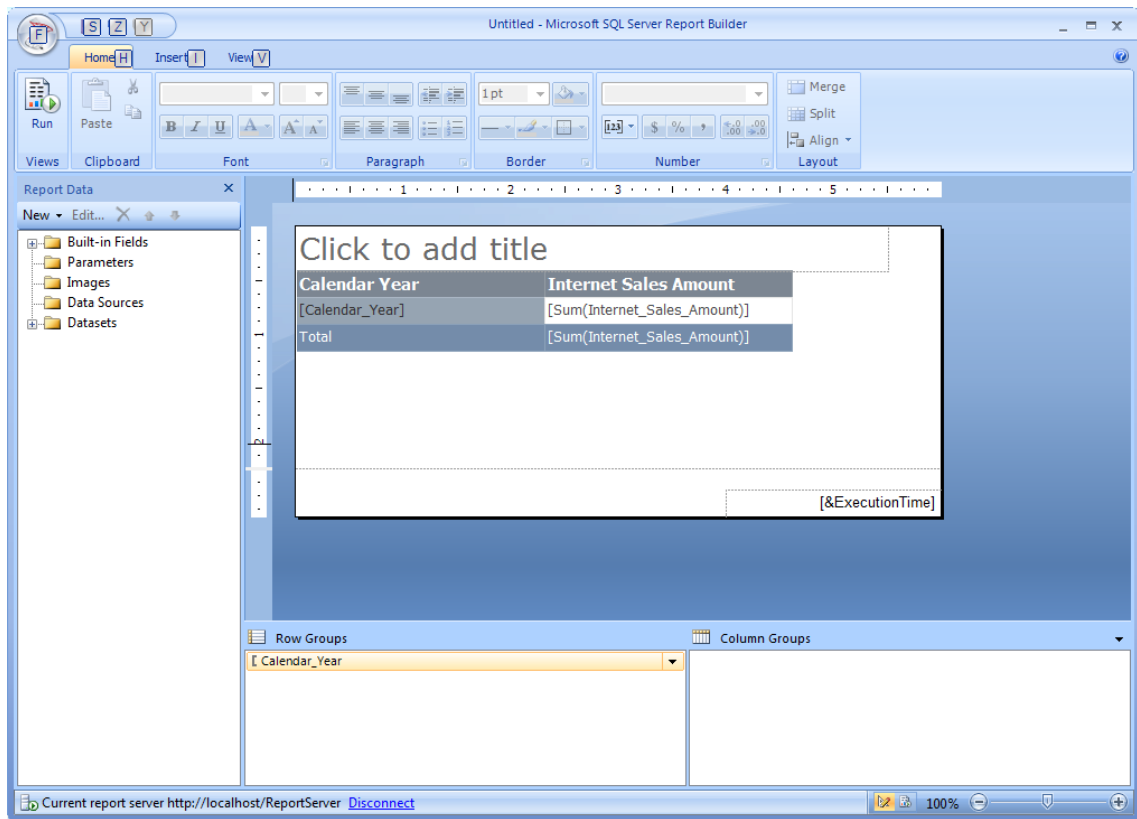


Figure 10–21. The SQL Server Report Builder GUI

You may also add filters to your report. Add a filter to your report by expanding the Dataset node in the Report Data pane, and expanding InternetSales. Right-click the Calendar_Year field, and select Field Properties. In the Dataset Properties dialog, select Filters to reveal the “Change filters” pane. Next, add and configure filter conditions that are appropriate for your business scenario.

Figure 10–22 shows the dialog box you use to add filters to your report, with 2005 selected as a calendar year filter. It’s interesting to see that the filters are generated from your “natural language” inputs. This is yet another time-saving feature built into this product. The idea is that business users can quickly and easily generate filters using natural language rather than having T-SQL or MDX programmers writing code.

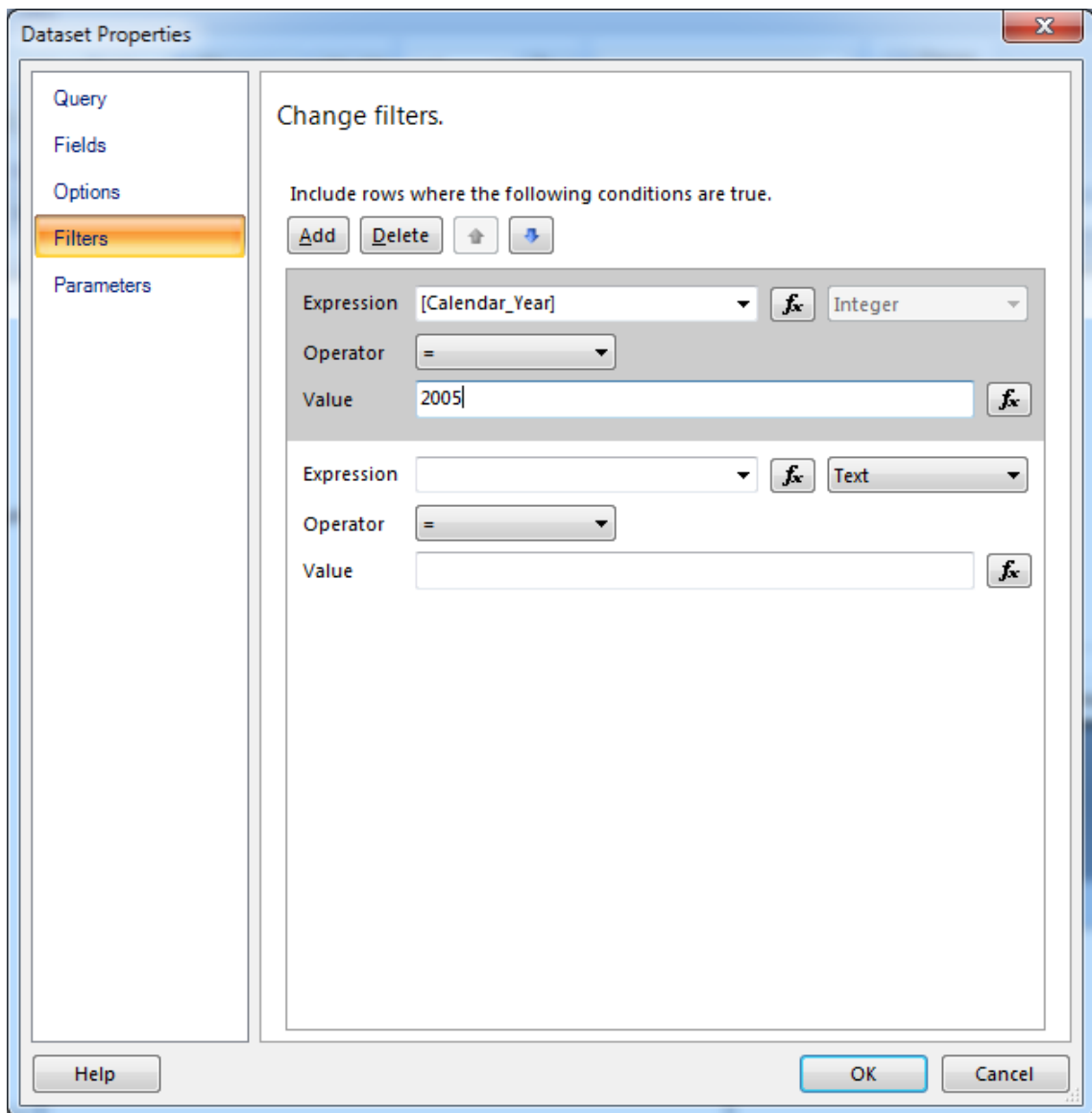


Figure 10–22. The Filter Data dialog box allows you to add filters to your report.

After filtering your report data, you may want to sort it as well. To sort your report data, right-click a column header, and select Tablix Properties, followed by Sorting. The Sorting dialog box is shown in Figure 10–23. Use it to add a sort on `Calendar_Year` in the same manner that you added the filter.

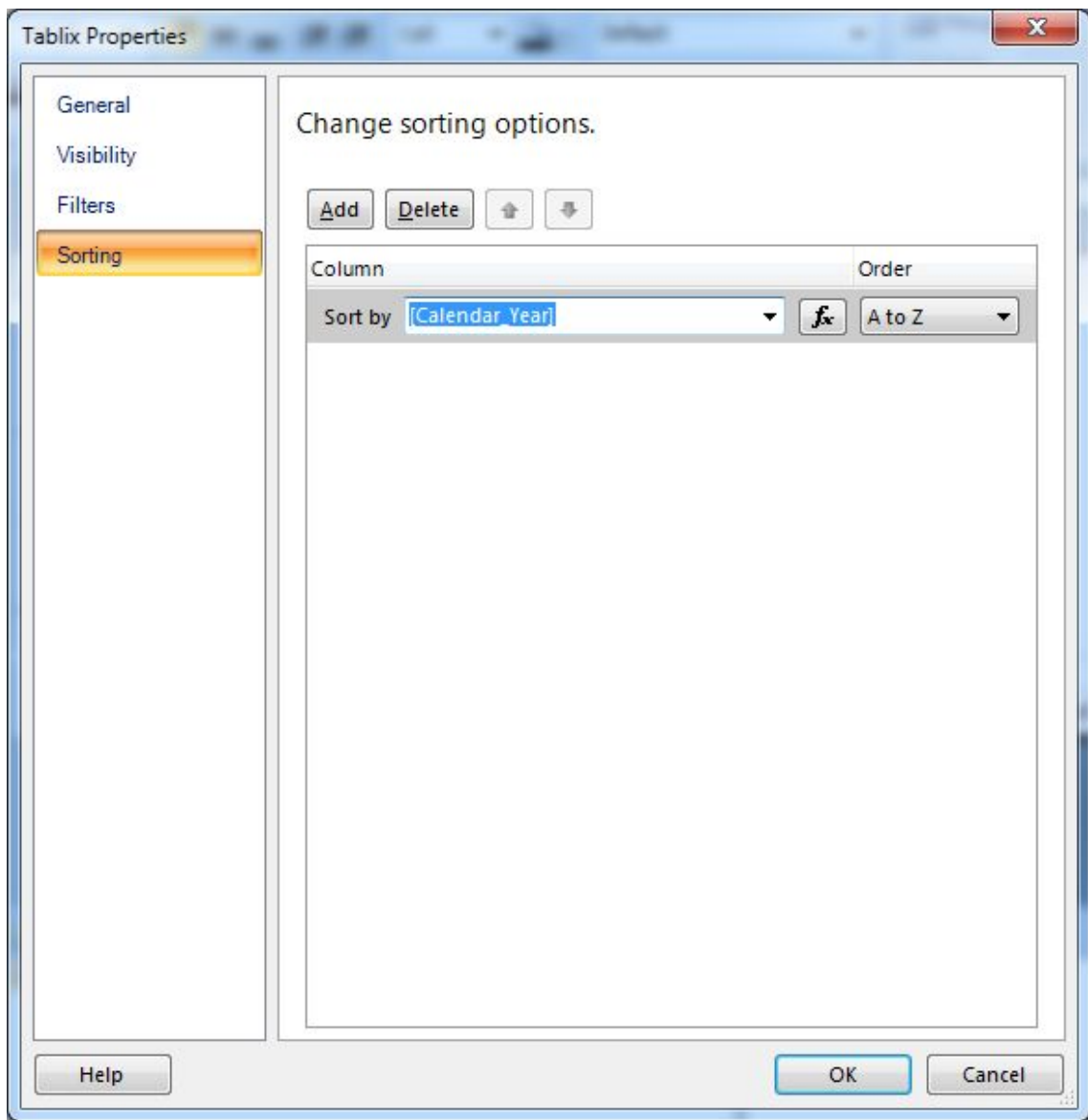


Figure 10–23. The Sorting dialog box allows you to sort the data displayed in your report.

You may also add a calculated field to your report by right-clicking your InternetSales dataset, and choosing Add Calculated Field. Click the Add button, select Calculated Field to add a row for your calculation, and click the “fx” button. The Expression dialog box appears; it lists the currently available entities and fields and a subset of functions that are appropriate to the particular selection, along with a description and example, as shown in Figure 10–24.

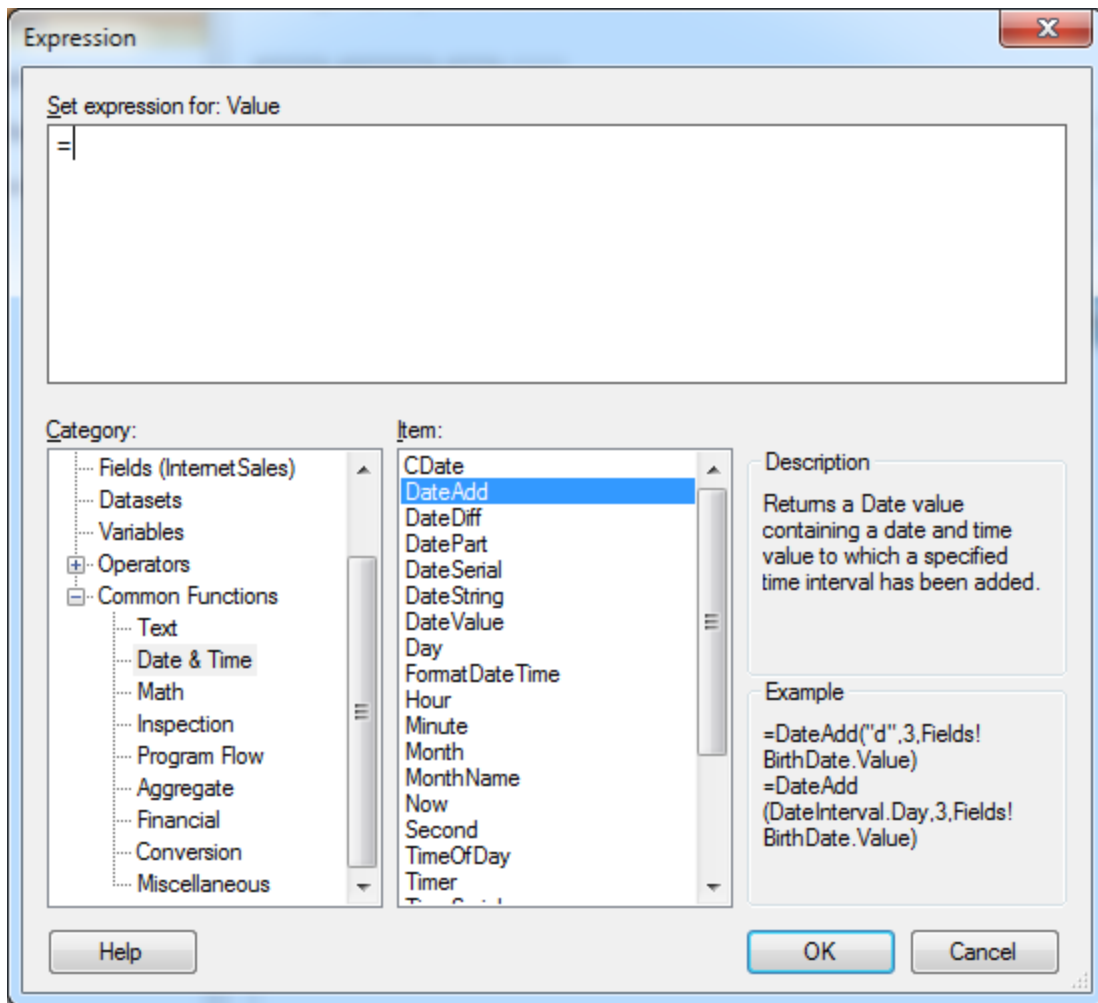


Figure 10–24. The Expression dialog box allows you to add a calculated field to your report.

The last area for you to configure prior to running your report is accessed by right-clicking outside the report page area and selecting Report Properties. In the Report Properties dialog box, you can configure other properties, including Page Setup, Code, References, and Variables. Figure 10–25 shows the Report Properties dialog box.

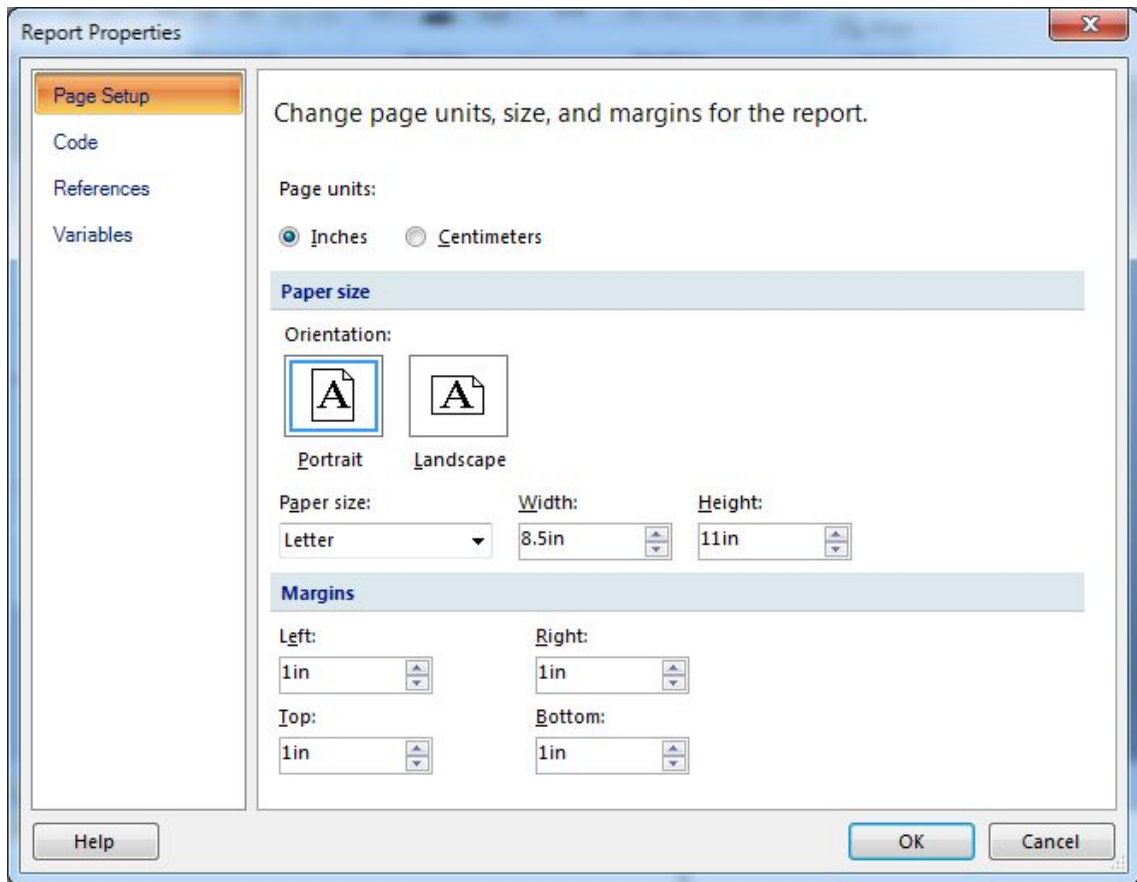


Figure 10–25. The Report Properties dialog box allows you to set a number of options for your report.

After you've configured your report, you may view it by clicking the Run button on the toolbar. If you want to create a report that will be viewable after this session (by you and anyone else to whom you grant permission) using Report Builder, you may save the report definition to the report server (provided you have the appropriate permissions). When you click the Radial menu and select Save, you are prompted to name your report and are directed to the `http://localhost/ReportServer` location by default; Figure 10–26 shows this dialog box. Any report that you build and save can be accessed using Report Manager.

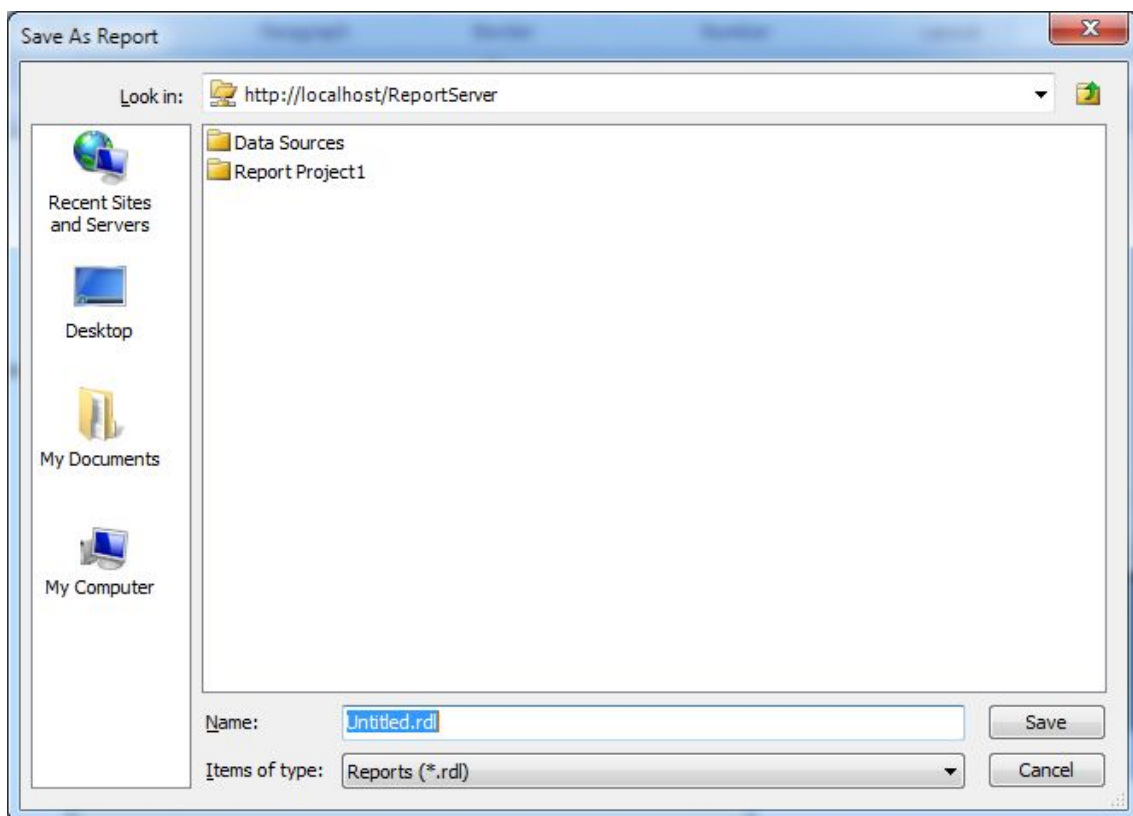


Figure 10–26. You have the option to save your report definition to the report server.

Report Builder is a great tool for you to build RDL reports against your SSAS cubes. Third-party vendor RDL generators are also available. An important consideration for your BI project is the determination of which tools will be used to create reports and which end users will be involved in report writing.

Summary

In this chapter, we reviewed capabilities of currently available client tools for SSAS. These included Excel pivot tables, pivot charts, SSRS, Report Manager, and the Report Builder. In the next chapter of the book, we'll discuss BI connectivity and reporting using Office 2010.



Data Mining with Excel

This chapter explores data mining with Excel, using both Excel workbooks and SSAS as data sources. Excel data mining is accomplished using the SQL Server 2008 Data Mining Add-in. Excel 2010 is a significant part of Microsoft's BI offering, and it provides you with an affordable and powerful end user client toolset. By offering such a rich feature set at such an attractive price, it is Microsoft's intent to make BI available to a larger percentage of your end user groups, rather than just the analyst community.

■ **Note** This chapter, along with Chapters 12 and 14, present an introduction to data mining and PowerPivot. Chapter 12 delves into the combined use of Excel and PowerPivot. Chapter 14 is an introduction to the methods and uses of data mining.

Exploring Excel 2010

The first significant enhancement Microsoft has made in Excel is a concerted effort to improve the end user client SSAS experience by improving the pivot table interface. The primary focus is to make pivot tables and charts more intuitive to set up and use, so that more of your end users will be able to quickly become productive using pivot tables.

The Excel Ribbon

Let's begin by taking a look at the new menu system. This menu redesign is called the *ribbon*. Figure 11-1 shows the first half of the Excel ribbon for the Data tab. You'll notice that this tab contains a group that allows you to manage connections to any data source to which you are making a connection. These connections include connecting from Excel to SSAS.

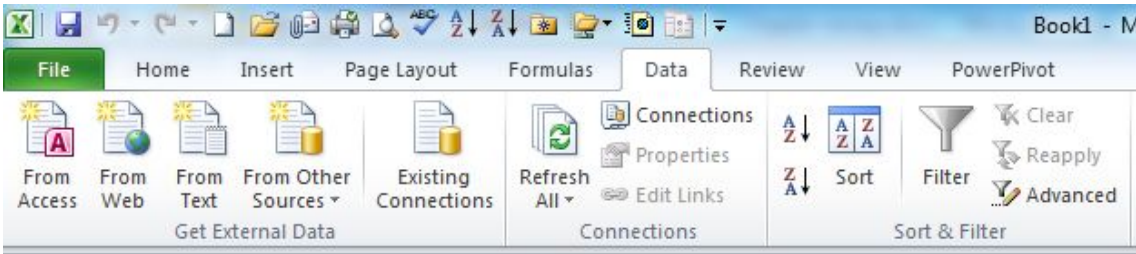


Figure 11-1. The left half of the Data tab in Excel

Another type of ribbon in Excel gives your end users quicker access to the pivot table menu items. This type of menu is called a contextual tabset, and it appears in the ribbon when an end user selects any area of an active pivot table by clicking it. The right half of the contextual tabset for pivot tables is shown in Figure 11-2.

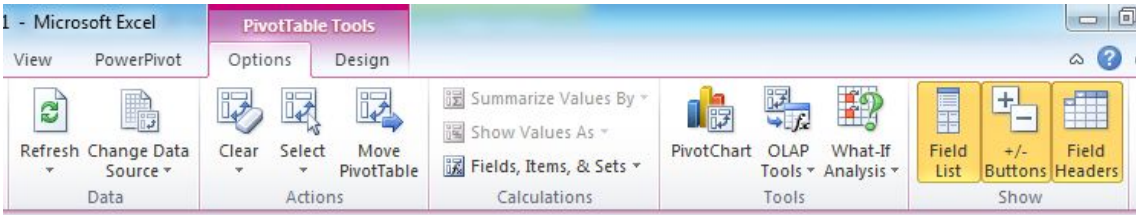


Figure 11-2. The contextual tab of the Excel PivotTable Tools ribbon

You can create a pivot table using SSAS source data via the ribbon menus. All menus are now available on a tab (for creating the initial connection to SSAS) or a contextual tab (for designing the pivot table) of the ribbon. The ribbon also simplifies access to some of the advanced SSAS features.

You can see another example of advanced SSAS feature support when you are using the Data Connection wizard. After you specify the name of the SSAS server and database that you want to connect to, you can then (in the next dialog box) select from a list of not only SSAS cubes but also SSAS perspectives for that particular SSAS database, as shown in Figure 11-3.

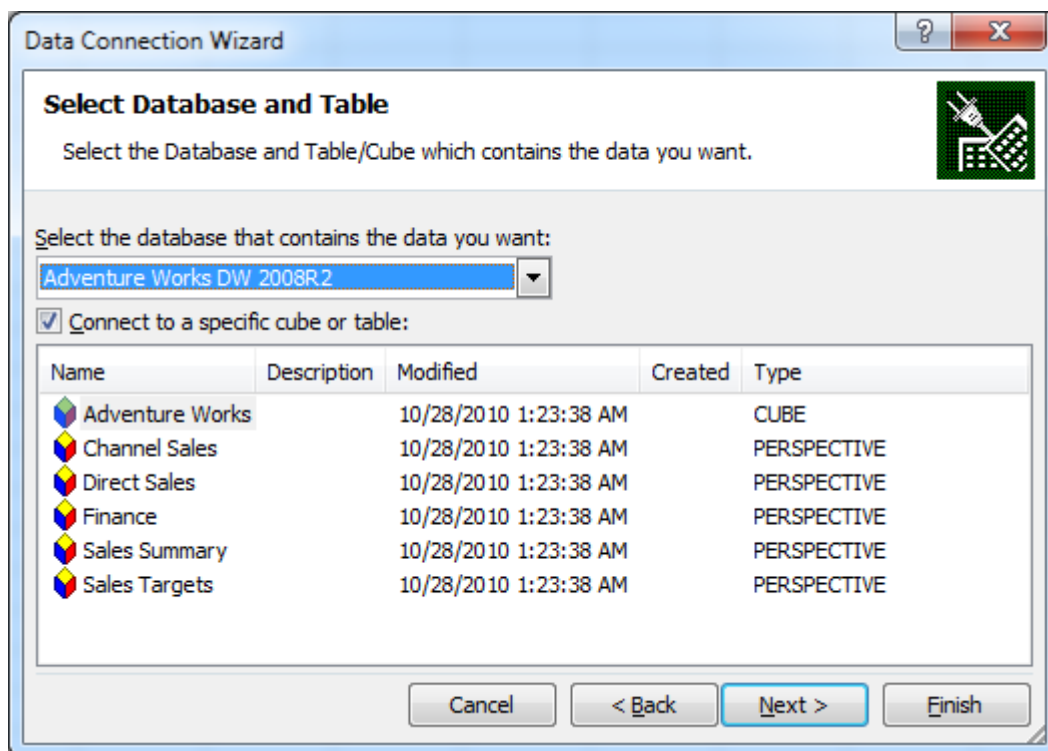


Figure 11–3. Associating your data connection with a specific cube or perspective

SSAS also supports cube drillthrough, and you can set the maximum number of records that can be retrieved via drillthrough on the Connection Properties Usage tab. After you've successfully connected to your SSAS instance, to quickly open the pivot table designer, you can simply click the Existing Connections icon on the Get External Data group of the Data tab. Next, in the Existing Connections dialog box, click the AdventureWorks item in the Connections in this workbook section, and then click Open, as shown in Figure 11–4.

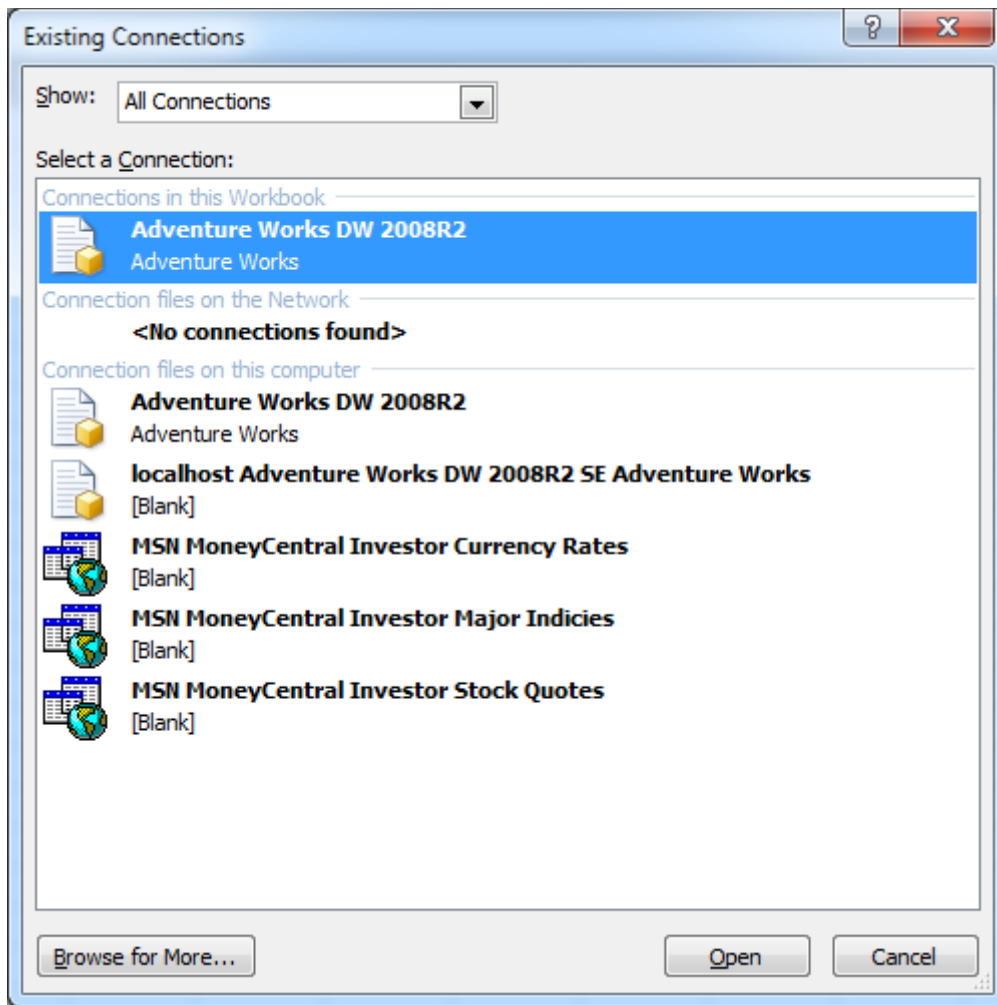


Figure 11–4. One way to create a pivot table is by opening a connection to an SSAS database.

You are then presented with an Import Data dialog box, which is set by default to PivotTable Report. You can also configure the Advanced properties of the connection by clicking the Properties button in the Import Data dialog box. Click OK to complete your connection configuration, and then Excel will open the pivot table design surface. The pivot table work area is more intuitive than previous versions, including “hint” text on the design surface and an intelligent PivotTable Field List. This list allows the user to drag and drop any field from the data source; for SSAS cubes, these fields will be measures, dimensions, and KPIs. The fields can either be dragged to the pivot table design surface or to the new descriptive boxes, called *areas*, below the PivotTable Field List. Figure 11–5 shows the default (blank) pivot table design surface and the new PivotTable Field List pane.

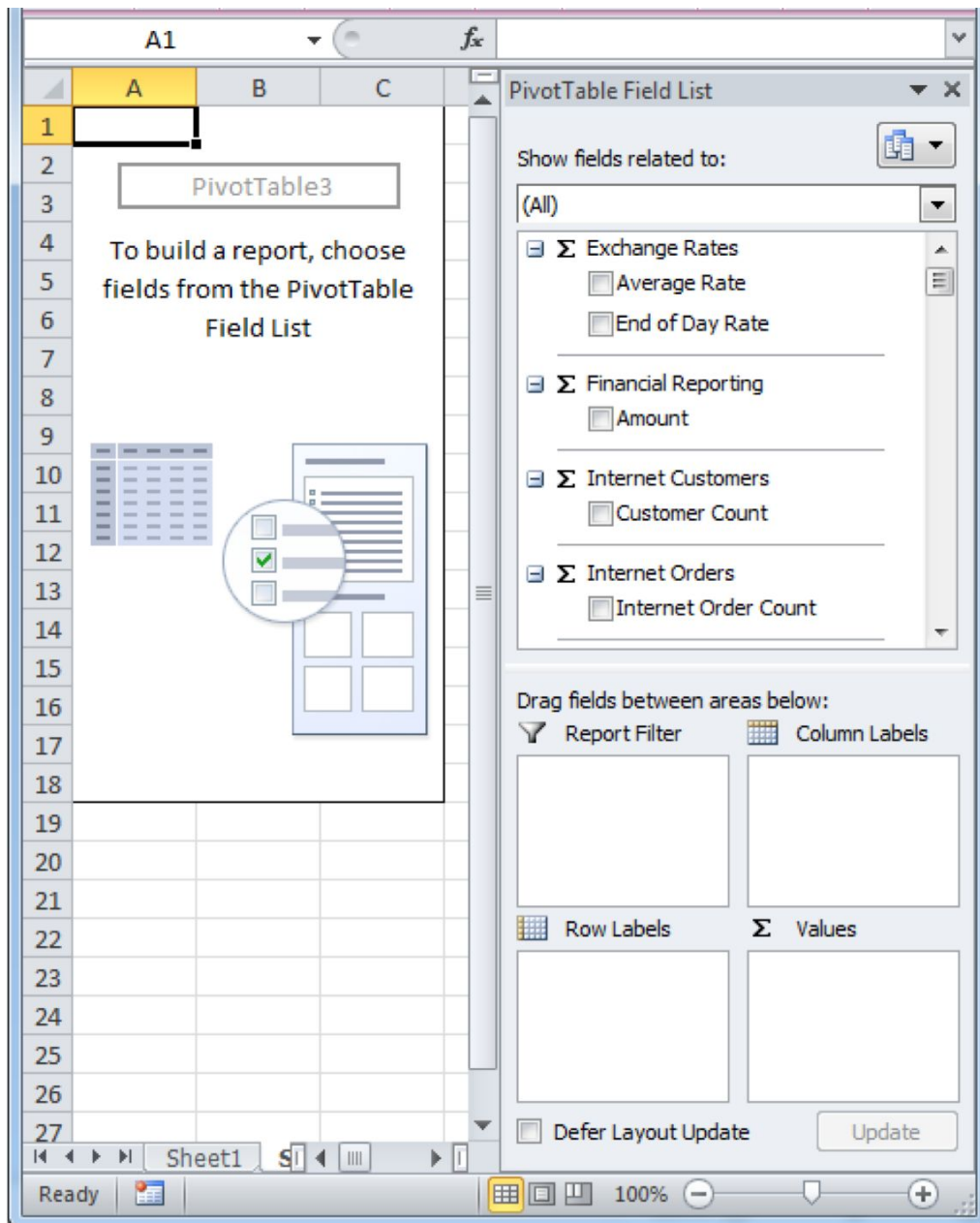


Figure 11–5. The pivot table design surface and PivotTable Field List improve the design experience for end users.

KPI Support in Excel

In addition to an intuitive design area, you'll note that SSAS KPIs are displayed as part of the fields in the PivotTable Field List. If you drag KPIs onto the pivot table design area, you'll also see that the associated icons are displayed in the pivot table. You'll probably find yourself using KPIs in a number of business scenarios, so be sure to include planning for KPIs in the business requirements phase of your BI project.

An example of the display of SSAS cube KPIs in an Excel workbook is shown in Figure 11–6.

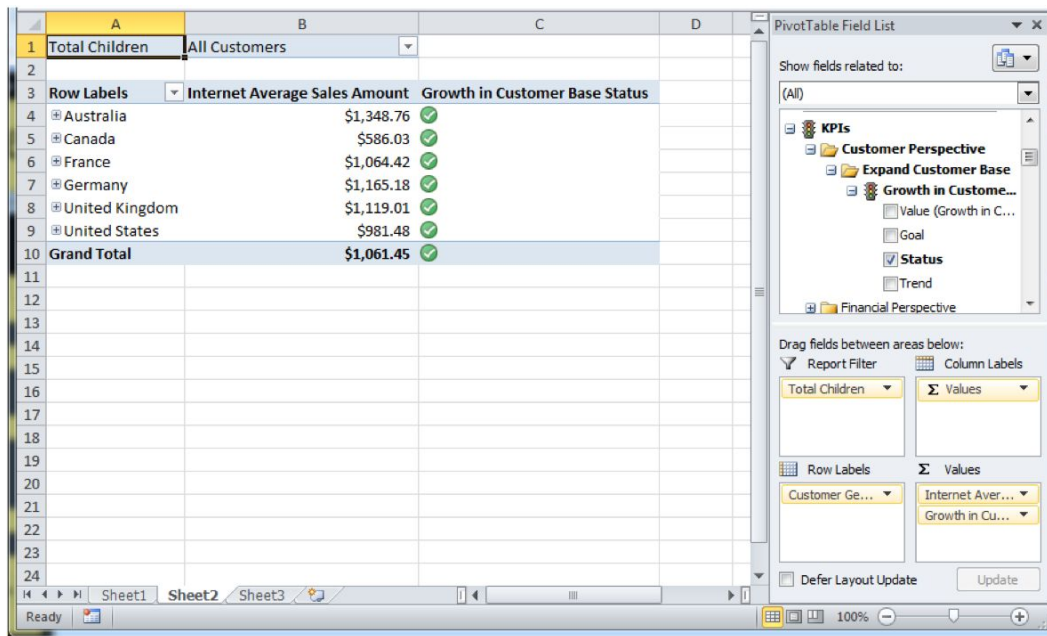


Figure 11–6. You can display KPI status indicator icons in your Excel pivot tables.

Take note that the Report Filter area on the PivotTable Field List allows you to easily work with a large number of filter fields for your report. Also, the field list has been set to show fields related to the Internet Average Sales measure only. Because most SSAS cubes will now be built using multiple fact tables, this ability to filter available fields based on fact names will be welcomed by end users.

In addition, Excel offers enhanced data visualizations. Although these features are *not* specific to pivot tables or to BI—in other words, you can apply them to any worksheet—you will find them to be particularly useful when working with the large result sets that can be returned from SSAS cubes. Most of these data visualization features are found on the ribbon's Home tab by clicking the Conditional Formatting button.

One example of using these new data visualization features is that you can use the built-in data bars to give a visual clue to values in your pivot table. To do so, simply select the cells from the pivot table that you want to apply the conditional formatting to, click the Home tab, click the Conditional Formatting button to expose the Data Bars button, and then click the color scheme you want to apply. You can also make your own custom scheme by clicking the More Rules button at the bottom of the supplied Data Bar Rules under the Conditional Formatting button on the Home tab of the ribbon. The result of applying this type of conditional formatting and using the Conditional Formatting button menu is shown in Figure 11–7.

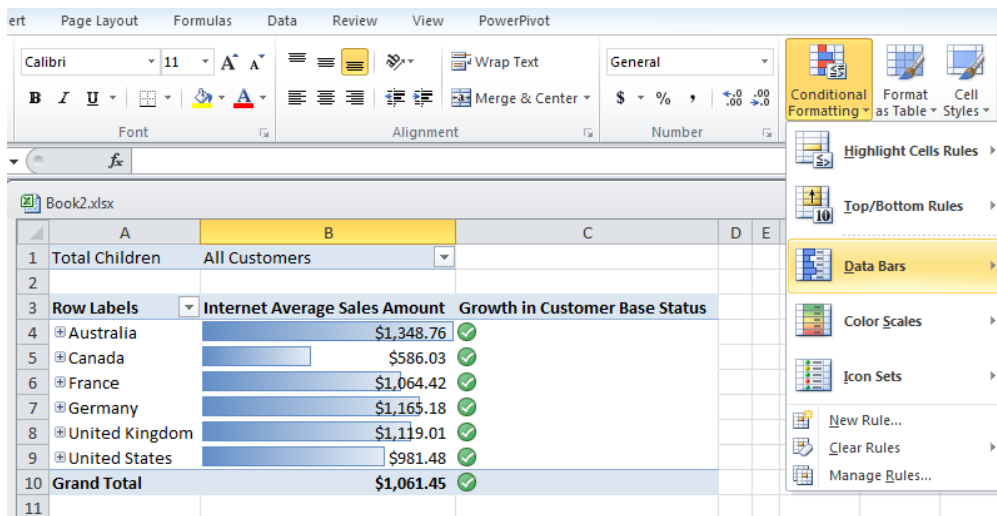


Figure 11–7. Using Data Bars Conditional Formatting can improve the visualization of your pivot table data.

Other types of data visualization tools included in Excel are Color Scales and Icon Sets. All of these conditional formatting schemes can be customized quite easily. Figure 11–8 shows the application of one of the default Icon Sets to the sample pivot table.

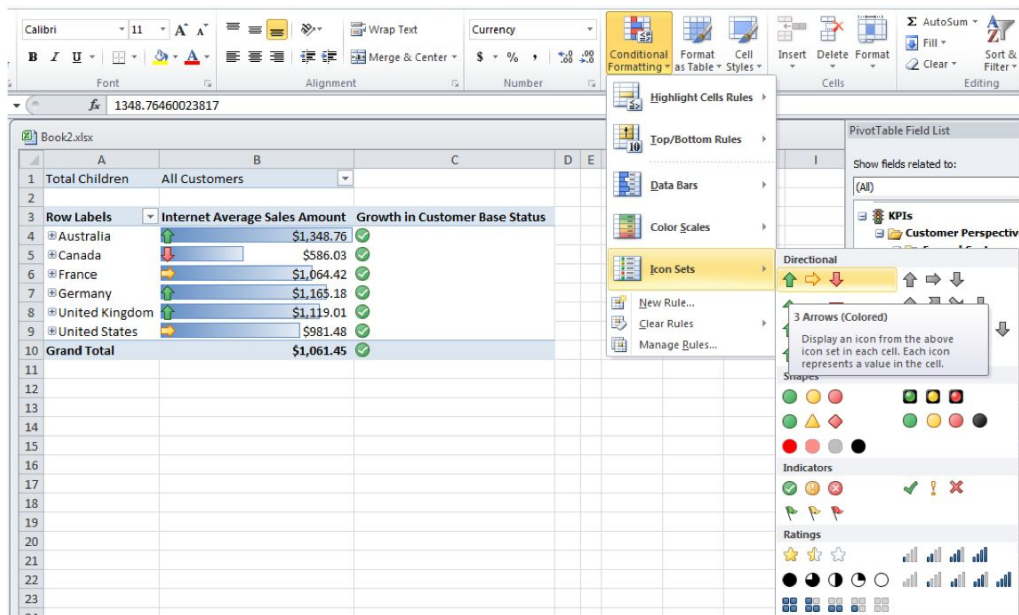


Figure 11–8. Applying built-in Icon Set Conditional Formatting to jazz up your pivot table

Pivot charts offer the ability to easily add a trend line. To do this, create a column chart from your pivot table by clicking the Pivot Chart button on the ribbon. Then right-click any of the data bars in the chart, and click Create Trendline. The line is created, and you can access the Format Trendline dialog box. A pivot chart with a trendline and the Format Trendline dialog box are shown in Figure 11–9.

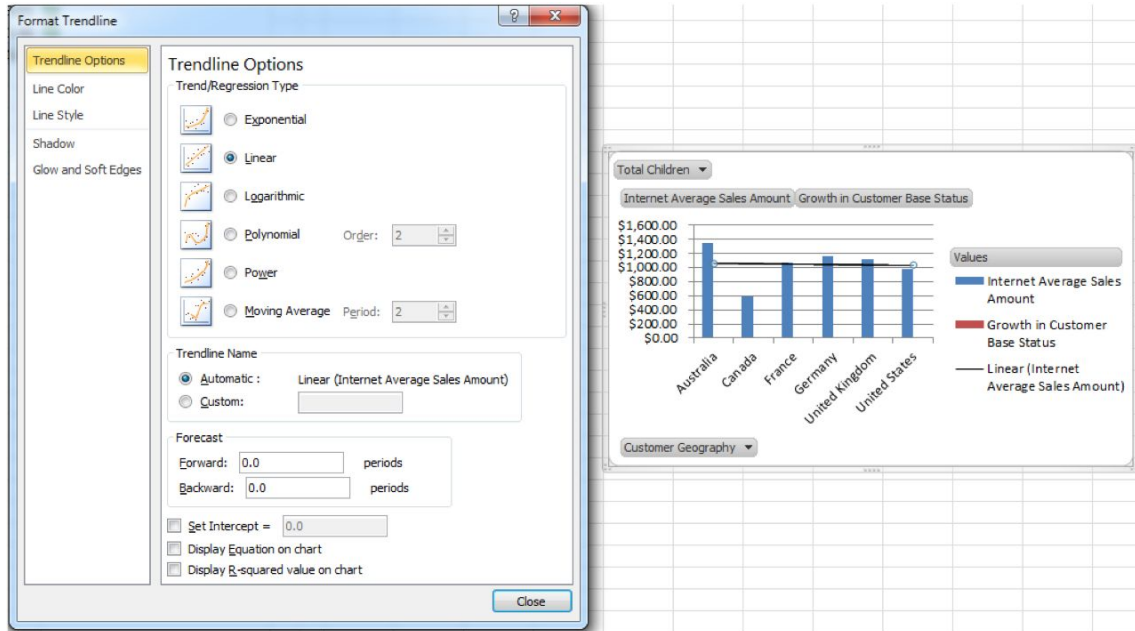


Figure 11–9. A pivot chart showing a trend line alongside the Format Trendline dialog box

Using Excel for Data Mining

One exciting feature in Excel is the ability to work as a native client for SSAS data structures. The next few sections will explore using Excel as a data mining client. To make this functionality available to Excel, you first need to install the SQL Server Data Mining Add-ins. The Data Mining Add-in is available as a download for Office 2007. However, it works with 32-bit Office 2010 as well. The add-in is accessible at www.microsoft.com/downloads/en/details.aspx?FamilyId=896A493A-2502-4795-94AE-E00632BA6DE7&displaylang=en.

Note The Data Mining Add-ins add functionality to both Excel and Visio as a 32-bit version only. There is currently no 64-bit version available. We will be reviewing the add-in functionality for Excel in this book.

Configuring Excel as a Data Mining Client

After installing the downloaded files using the Setup wizard, run the Getting Started wizard by choosing Getting Started from the Microsoft SQL 2008 Data Mining Add-ins program group, show in Figure 11–10.

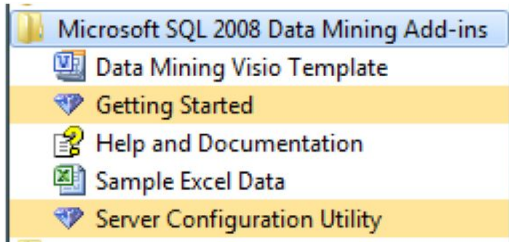


Figure 11–10. The Data Mining Add-ins group of the Start menu

The Getting Started wizard contains a good amount of documentation on each page, making it easy to use. Figure 11–11 shows the first page of the wizard.

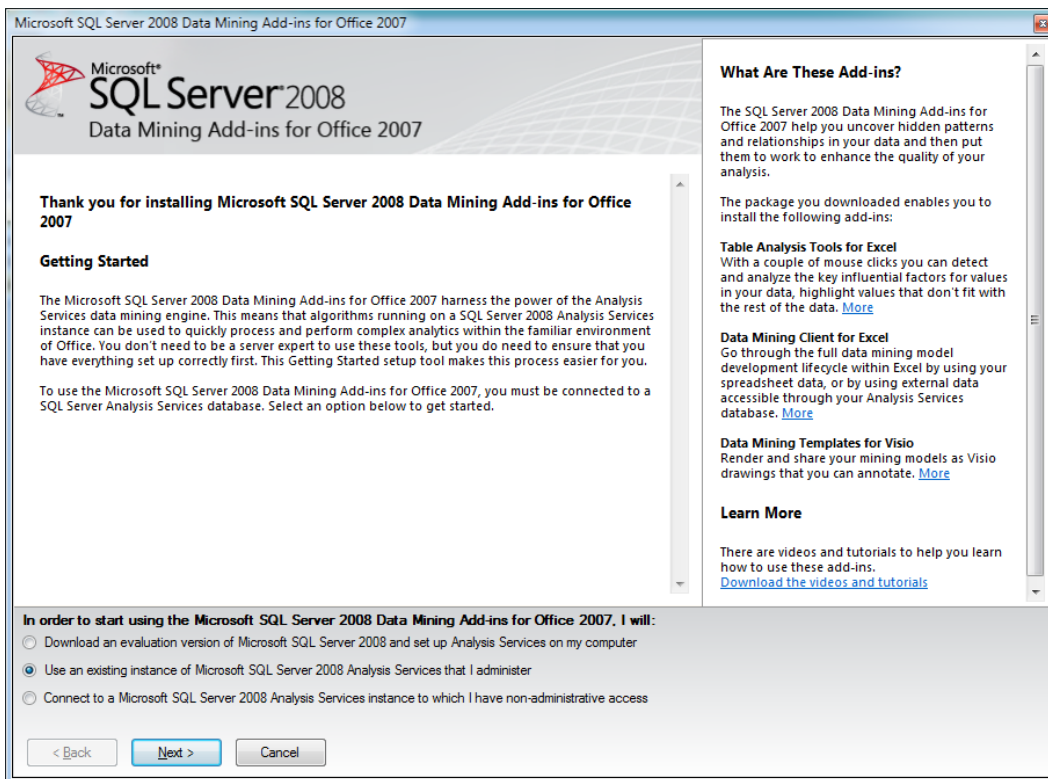


Figure 11–11. The Data Mining Add-ins Getting Started wizard

On the first page of the Getting Started wizard, select the “Use an existing instance of Microsoft SQL Server 2008 Analysis Services that I administer” radio button, and click Next. Execute the Server Configuration Utility by clicking the link on the second page of the Getting Started wizard. When the “Welcome to the SQL Server 2008 Data Mining Add-in Configuration Wizard” dialog appears, click Next. In the first step of the configuration wizard, specify the SSAS server name. In step two, specify that you want to allow the creation of temporary mining models, as shown in Figure 11–12.

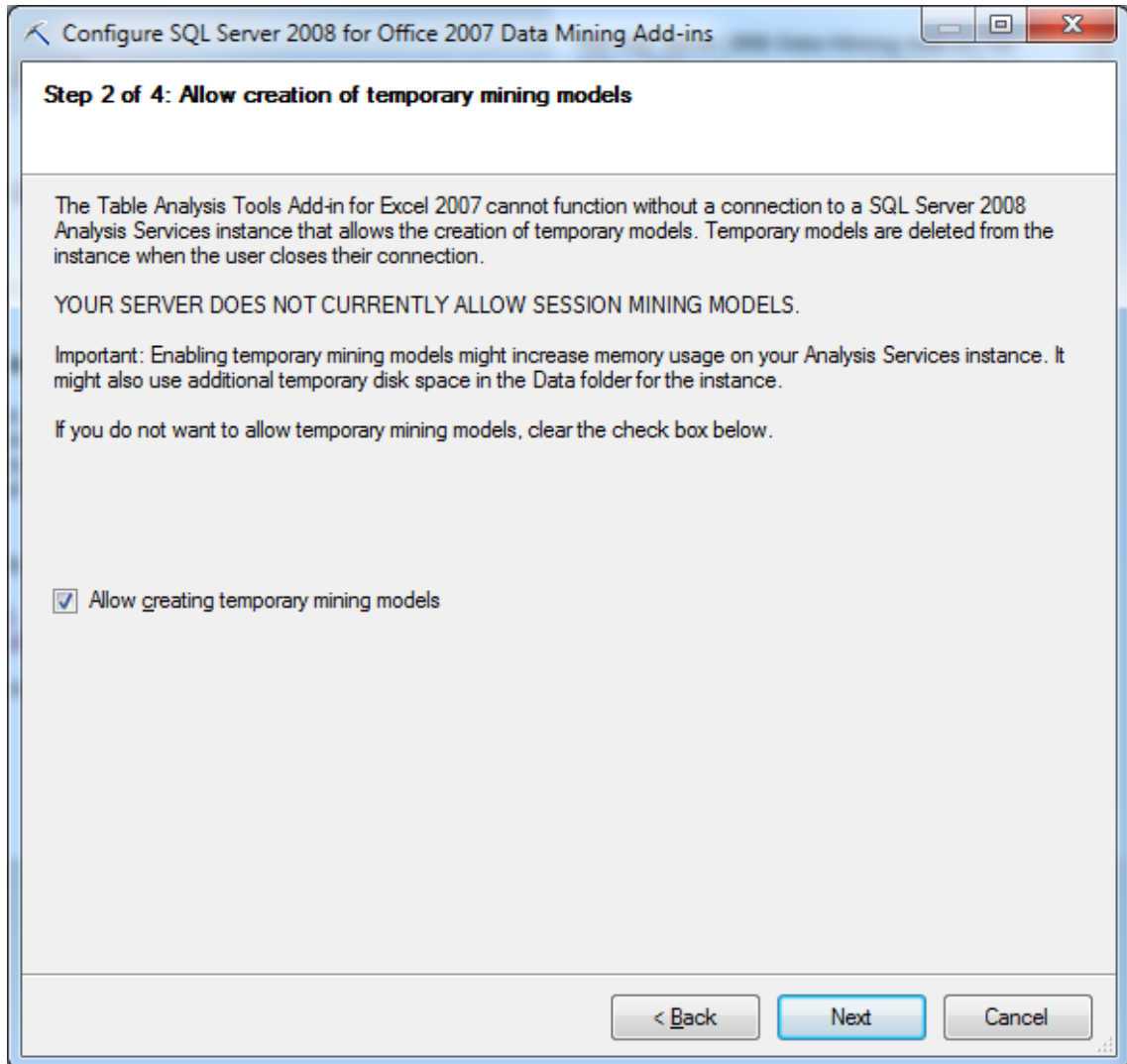


Figure 11–12. Determining your user’s ability to create temporary mining models

In step three, create a new database called DMAddinsDB. Creation of temporary mining models gives your users the flexibility to freely create models while allowing you to manage permissions for these users without impacting other databases. In the last dialog box of the wizard, grant add-in users appropriate permissions, as shown in Figure 11–13. As with the other dialog boxes in this wizard, an explanation of the implications of the task you are performing is given on the dialog box. You should, of course, implement security settings based on your business requirements. This usually includes, at a minimum, restricting the ability to create permanent models to the smallest possible subset of end users.

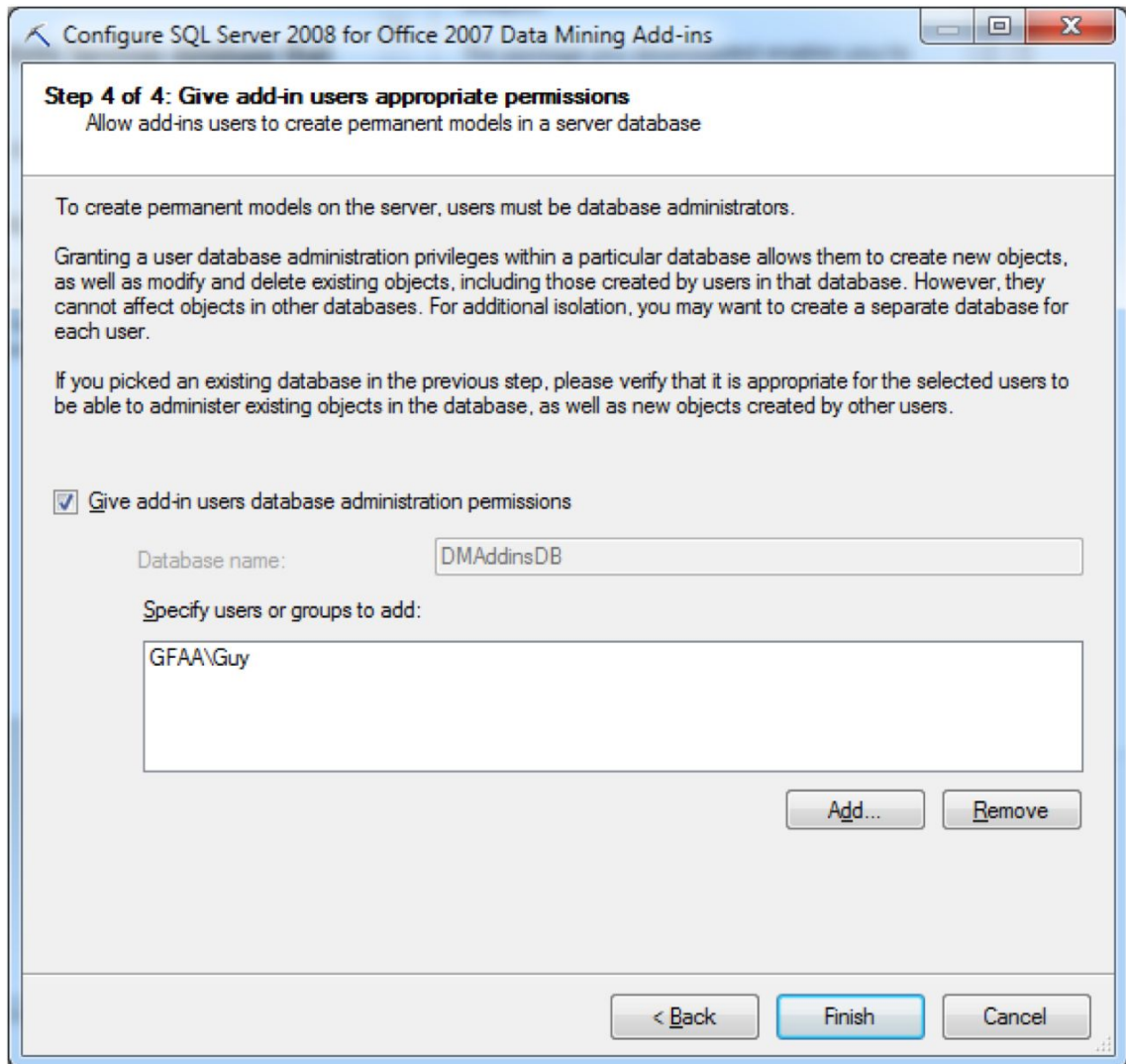


Figure 11–13. Setting user permissions for accessing data mining models

As the wizard completes, it presents you with a status dialog box, showing the success (or failure) of each of the required setup steps. If you have any failures, simply rerun the wizard. The most common cause of failure is an incorrectly configured connection string. This status dialog box is shown in Figure 11–14.



Figure 11–14. Confirmation of successful configuration

Using Excel as a Data Mining Client

The best way to start using Excel as a data mining client for existing SSAS data mining structures and models is to explore the various activities available on the Data Mining tab, which is added to Excel's ribbon when you install the Data Mining Add-ins. Let's start by examining how to connect to and browse mining models. This section of the ribbon is shown in Figure 11–15.

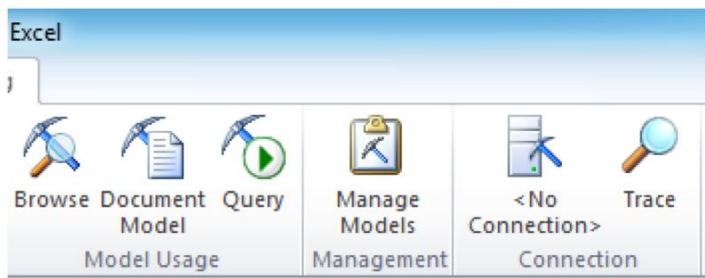


Figure 11–15. The Data Mining tab groups: Model Usage, Management, and Connection

The Connection group of the ribbon allows you to add, edit, delete, test, or make current one or more connections to SSAS. Click No Connection in the Connection group to open the Analysis Services Connections dialog, and click New. In the Connect to Analysis Services dialog that appears next, enter your server name, and select Adventure Works DW 2008R2 as the catalog to connect to. Click OK, and then Close to create the new connection.

Using the Manage Models button in the Management group, you can view metadata about data mining structures and models. You can also perform administrative actions such as renaming, clearing, and processing the structure or model. The Managing Mining Structures and Models dialog box is shown in Figure 11–16.

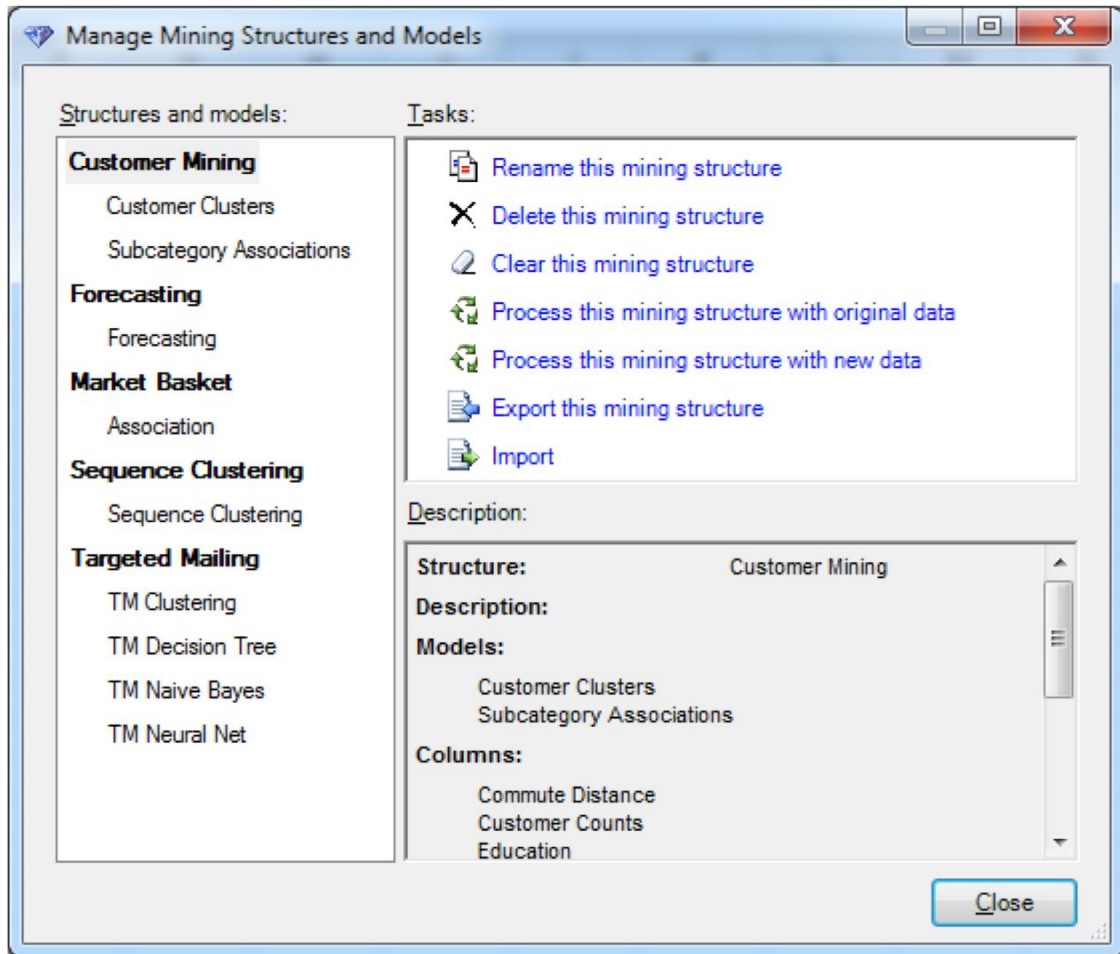


Figure 11–16. The Manage Mining Structures and Models dialog

The Browse button on the Model Usage group allows you to view the mining models in the structures of the SSAS database you've connected to. For example, if you view a model created using the

Naïve Bayes algorithm, you'll be shown the Dependency Network Viewer. The type of viewer available depends on the type of mining model algorithm selected to build the original model.

Note The Naïve Bayes data mining algorithm is a simple classifier of data. It is called Naïve because it gives equal weight to all data points that contribute to a classification created by the algorithm. Naïve Bayes and the Dependency Network Viewer are covered in more detail in Chapter 14.

When you click the Browse button, you'll be presented with a dialog box listing all of the mining models, arranged by algorithm type, that are associated with your particular mining structure. After you click a particular model to select it and then click Next, you'll be presented with the associated Mining Model Viewer or Viewers. In the Browse window, you can manipulate the mining model using the same techniques that you used in BIDS or SSMS. Figure 11-17 shows a sample Browse window in Excel. Note that you have the option to copy the information in the Browse window to Excel. If you choose to copy to Excel, the model will be rendered in a way that is "friendly" to Excel. The render method used by Excel varies depending on the type of Mining Viewer. Figure 11-18 shows the output of saving the model shown in Figure 11-17.

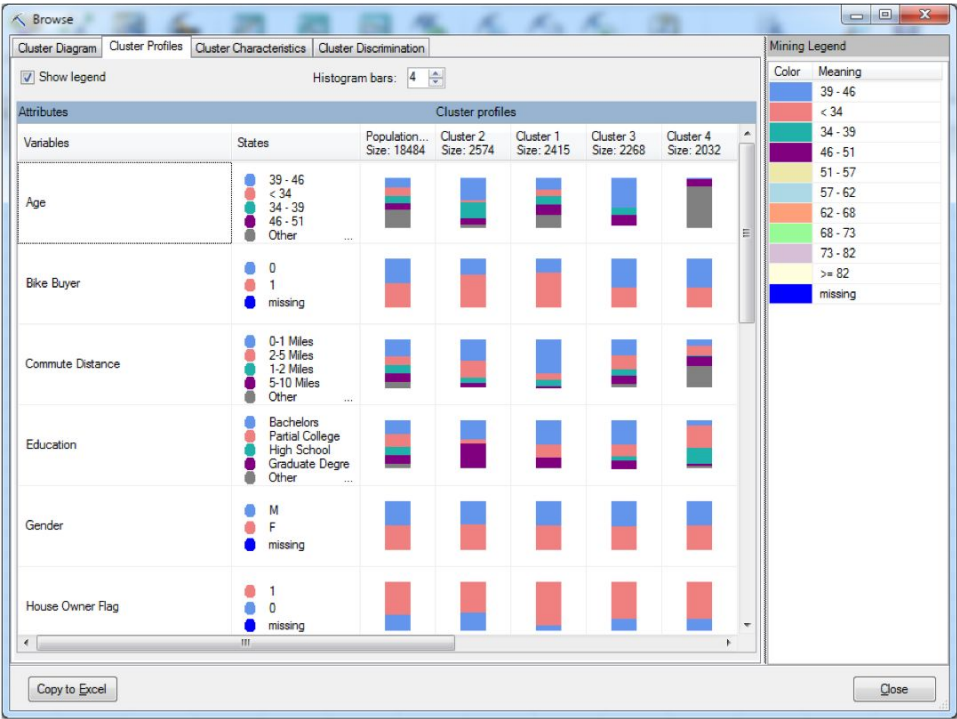


Figure 11-17. After you select a mining model to browse, Excel will display the model viewer types associated with that particular algorithm.

| TM Clustering | | | | | | | |
|------------------|---------------------|------------------|-----------|-----------|-----------|-----------|-----------|
| Cluster Profiles | | | | | | | |
| Variables | States | Population (All) | Cluster 2 | Cluster 1 | Cluster 3 | Cluster 4 | Cluster 5 |
| Size | | 18484 | 2574 | 2415 | 2268 | 2032 | 1980 |
| Age | 39 - 46 | 3896 | 46 % | 24 % | 60 % | 2 % | 5 % |
| Age | < 34 | 3519 | 5 % | 13 % | 0 % | 0 % | 0 % |
| Age | 34 - 39 | 2934 | 33 % | 18 % | 16 % | 0 % | 1 % |
| Age | 46 - 51 | 2552 | 15 % | 22 % | 23 % | 15 % | 20 % |
| Age | 51 - 57 | 2262 | 1 % | 14 % | 1 % | 38 % | 36 % |
| Age | 57 - 62 | 1907 | 0 % | 6 % | 0 % | 33 % | 28 % |
| Age | 62 - 68 | 1003 | 0 % | 2 % | 0 % | 10 % | 10 % |
| Age | 68 - 73 | 343 | 0 % | 0 % | 0 % | 1 % | 1 % |
| Age | ... | ... | ... | ... | ... | ... | ... |
| Bike Buyer | 0 | 9352 | 34 % | 28 % | 58 % | 60 % | 58 % |
| Bike Buyer | 1 | 9132 | 66 % | 72 % | 42 % | 40 % | 42 % |
| Commute Distance | 0-1 Miles | 6310 | 43 % | 69 % | 32 % | 14 % | 9 % |
| Commute Distance | 2-5 Miles | 3234 | 35 % | 13 % | 29 % | 20 % | 4 % |
| Commute Distance | 1-2 Miles | 3232 | 12 % | 13 % | 14 % | 3 % | 39 % |
| Commute Distance | 5-10 Miles | 3214 | 10 % | 5 % | 18 % | 19 % | 41 % |
| Commute Distance | 10+ Miles | 2494 | 0 % | 0 % | 7 % | 44 % | 8 % |
| Education | Bachelors | 5356 | 40 % | 50 % | 51 % | 12 % | 0 % |
| Education | Partial College | 5064 | 9 % | 26 % | 24 % | 46 % | 22 % |
| Education | High School | 3294 | 0 % | 1 % | 9 % | 32 % | 44 % |
| Education | Graduate Degree | 3189 | 51 % | 23 % | 17 % | 5 % | 0 % |
| Education | Partial High School | 1581 | 0 % | 0 % | 0 % | 5 % | 34 % |

Figure 11–18. Copying the information from the Mining Model Viewer to Excel using the Copy to Excel button

The Query button allows you to execute a DMX prediction query against a mining model. There are two modes in which to work: basic or advanced. As with many of the other Data Mining Wizard tools, the first page of this wizard provides an explanation of what you can accomplish by using it.

Note Chapter 14 will give you a better understanding of this wizard's functionality. You will learn how to associate external data with an existing model, map columns, and create output based on these inputs.

Using data mining, you could use some competitor's data that you've purchased as external data to compare that competitor's sales results with the results of an existing model that reflects your results for a particular business scenario. The first page of the Query Model wizard is shown in Figure 11–19.

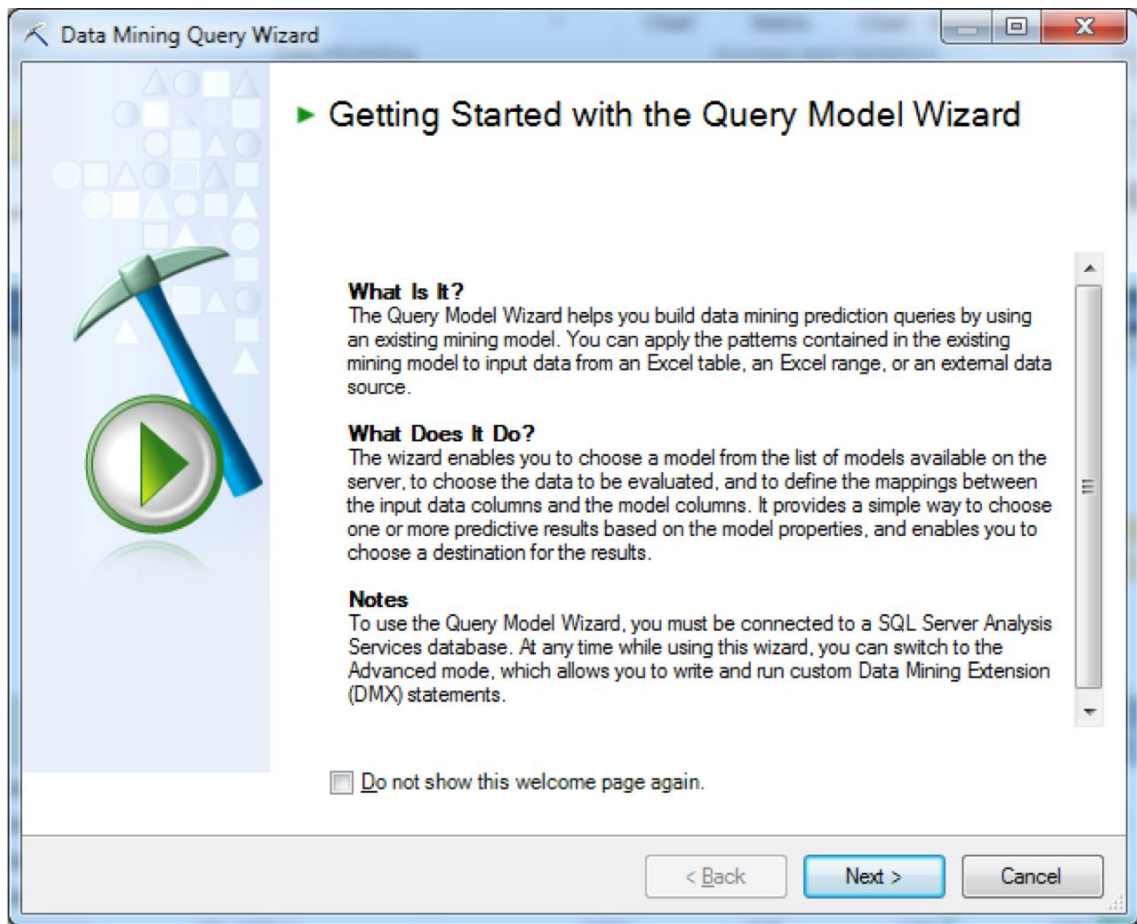


Figure 11–19. The Query Model wizard

The first step is to select the source model. Here you are presented with the same dialog box that you worked with when using the Browse button. After you select the source mining model, you'll be asked to select the input data for comparison. As stated on the first page of the Query Model wizard, you may choose from Excel tables, Excel ranges, or SSAS data sources for this data.

Also on this dialog box of the wizard, you have the option to move to the advanced view. If you do so, you'll have the ability to edit the query, add DMX templates, choose a new model, select input, map columns, or add output. This dialog box is shown in Figure 11–20.

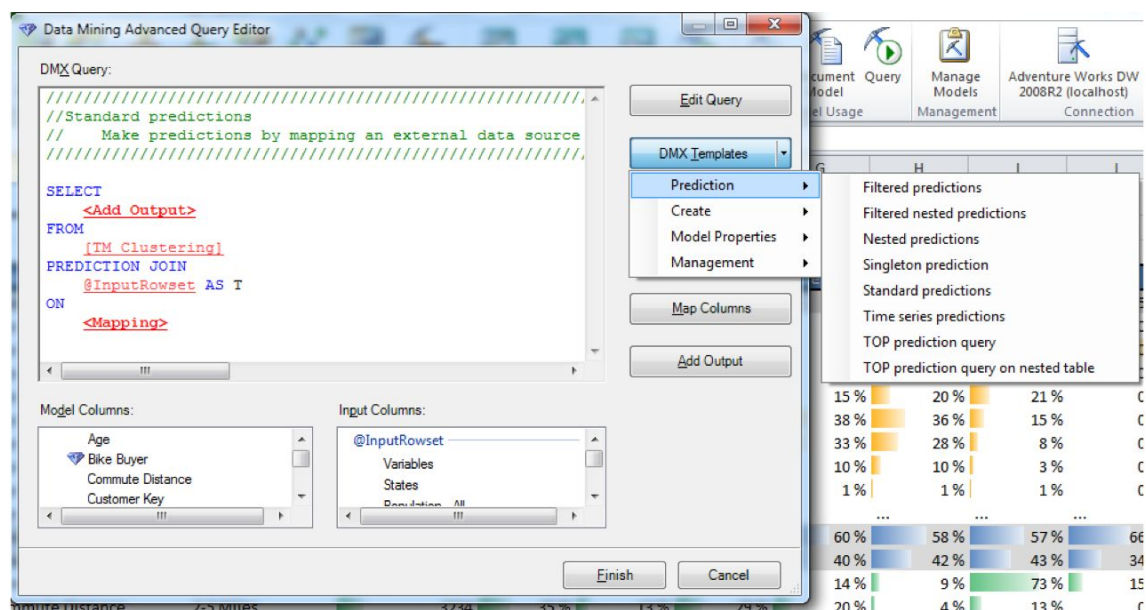


Figure 11–20. Using the Advanced Query Editor panes to modify a DMX query

Also in the Advanced Query Editor, you can view and modify the column mappings of the source mining model and new data. Then you configure the Add Output dialog box to specify the type of prediction query you want to execute. The default selection values are as follows:

- Predict: This function evaluates the numeric expression specified in Numeric_Expression in another data mining model.
- PredictProbability: This function returns the probability for a specific value or state. It applies to a scalar column and returns a scalar value.
- PredictSupport: This function returns the support value (that is, relevancy ranking) for a specific state.

Note The ideas of *specific value* and *state* are close related. For example, a bike buyer attribute may be predicted to have a specific value of 1. The state of this prediction is that of “Bike Buyer”.

You may also choose to display more values by selecting the Advanced button of the dialog box, as shown in Figure 11–21.

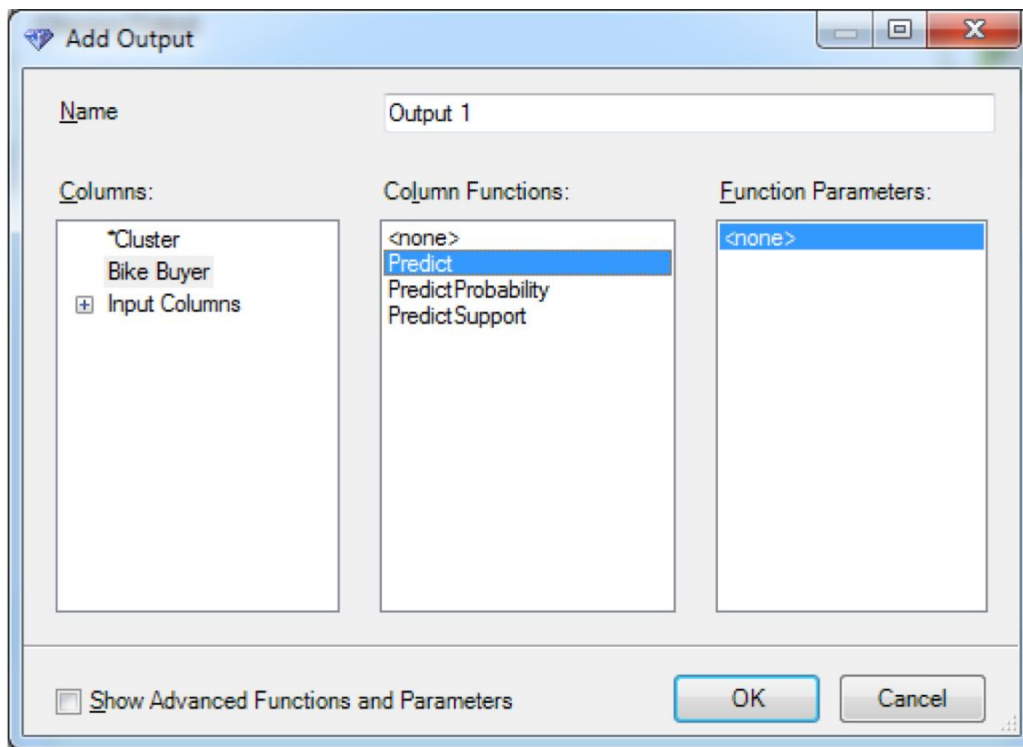


Figure 11–21. Adding output functions to your queries

Your final choice in the wizard is to determine where you want the output of the query placed. Your options are to append to the input data, create a new worksheet, or place in an existing worksheet.

Using the Data Preparation Group

This section of the Data Mining ribbon is primarily designed to allow Excel users to implement data mining techniques using Excel source data rather than SSAS data. Despite this limitation, the features are still worth exploring as your end users will probably find this functionality useful when they are performing data mining operations. Obviously, performing data mining operations on SSAS data provides the advantage of a centralized set of data to work with. There may be some situations, however, where data that is specific to a particular user (and stored in an Excel workbook) could be made more useful by performing data mining operations on it.

Explore Data, Clean Data, and Sample Data are the three buttons on the Data Preparation group of the Data Mining ribbon, as shown Figure 11–22.

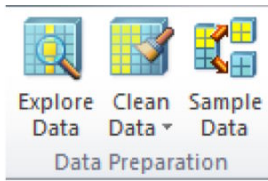


Figure 11–22. *The Data Preparation group of the Data Mining ribbon*

The first two buttons, Explore Data and Clean Data, are designed to work with Excel data only. They use data mining concepts to allow you to view and clean your data by removing a specified number of outliers (or exception cases) and/or by relabeling (or renaming) specified column values. Performing one, or both, of these operations can improve the usability of a set of data by removing or minimizing exceptions or distractions.

To use the Explore Data function, first select the table (worksheet) or data range in Excel. In the next page of the wizard, select the column for which you want to generate a visualization. The last page of the wizard produces a column chart by default; you may change this to a line chart by clicking that button on the dialog box. Figure 11–23 shows some sample output from the Explore Data button. The purpose of this view is to allow you to quickly visualize your data in a histogram format.

The Clean Data button, like Explore Data, is only designed to work with Excel data. There are two methods of data cleansing available: removing outliers and relabeling fields. *Outliers* are defined in data warehousing as exceptional cases; that is, those outside of a normal range. When you click the Clean Data button on the ribbon and select Outliers, the wizard is launched.

You perform the same setup steps as you did using the Explore Data wizard, such as selecting source data and so on. The difference in this wizard is in the Outliers dialog box. Here you specify the threshold for outlier removal. This dialog box shows you a visual preview of the effect of removing a variable number of outliers (see Figure 11–24).

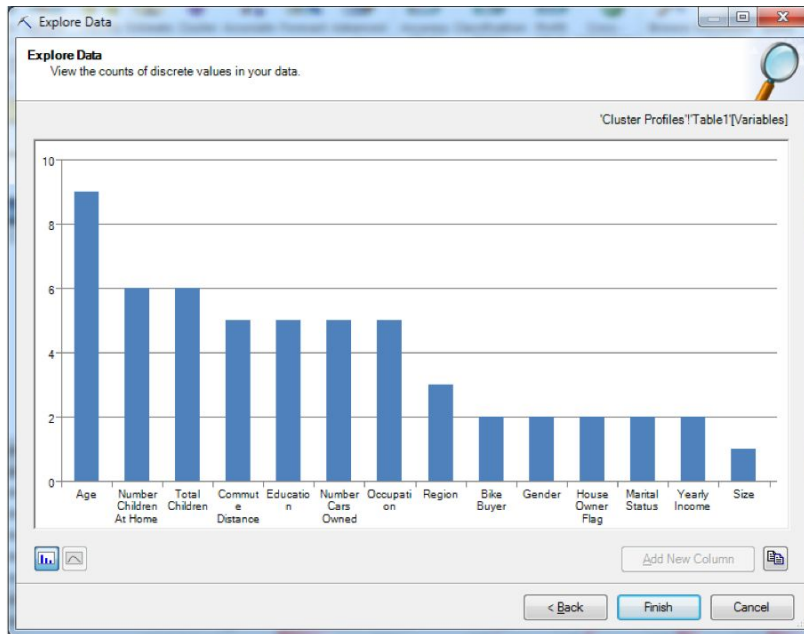


Figure 11-23. The Explore Data button allows you to quickly visualize Excel data.

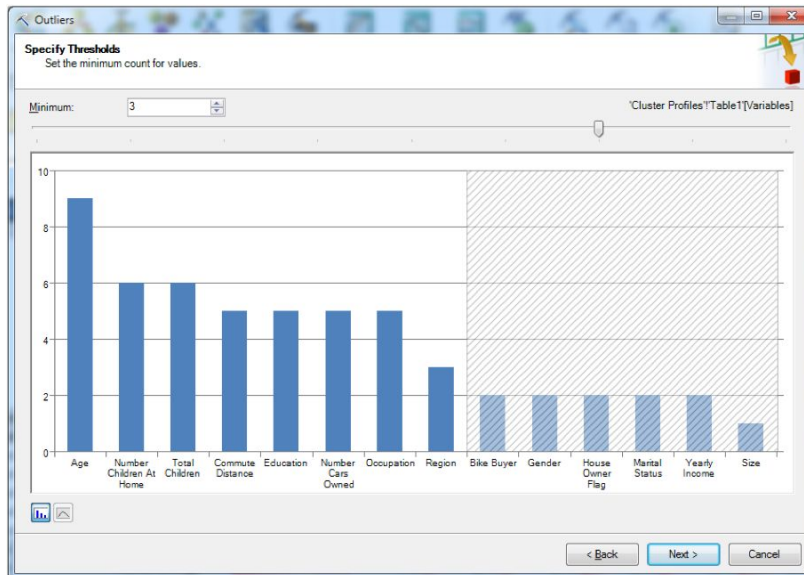


Figure 11-24. Specifying thresholds for removing outliers

The Sample Data button, unlike the other two, can be used with Excel or SSAS data. The first dialog box of the Sample Data wizard explains the functionality of this tool and is shown in Figure 11–25.

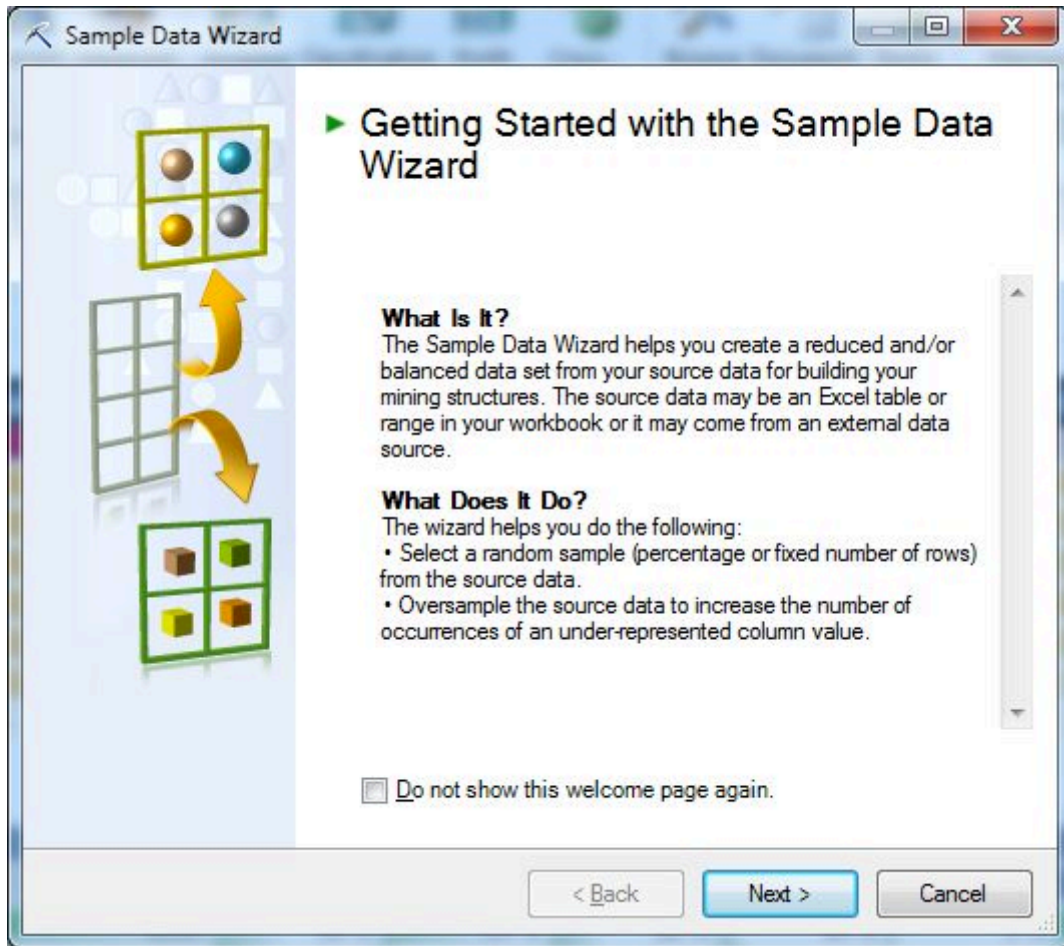


Figure 11–25. The first dialog box of the Sample Data wizard explains the wizard's functionality.

The Sample Data wizard allows you to sample your source data in one of three possible ways. For Excel data only, you can split your source data into two partitions so that you can use one set for data mining model training and the other setting for testing the validity of the mining model that you've created. You can also oversample columns from the source data to "balance" selected values in an underrepresented column to "smooth out" abnormalities in source data. If you are using either Excel or SSAS data as a source for the Sample Data wizard, this wizard allows you to conduct random sampling. This technique is used to reduce the size of a data set that is used for input to a data mining model. The technique is also used to facilitate rapid prototyping because data mining models based on smaller sized data sets process more quickly.

In the second dialog box of this wizard, you are asked to select the Excel worksheet, data range, or the SSAS data source. The next choice in the wizard is dependent on your previous selection. If you've selected SSAS, then you'll use the visual query designer to create the SSAS data source. After you complete this step, you'll be presented with the Select Sampling Type dialog box.

For SSAS queries, the only choice available is random sampling. The output of this choice is well-documented on this dialog box, as shown in Figure 11–26. As mentioned previously, the main purpose of this functionality is to reduce the size of an initial training sample. It is often used during the initial proof-of-concept or early rapid prototyping phases of an SSAS project to produce a small but structurally correct data source.

If you select Excel data as your source data for the Sample Data wizard, then all sampling types are available via the wizard: Random sampling or Oversampling to balance data distributions options.

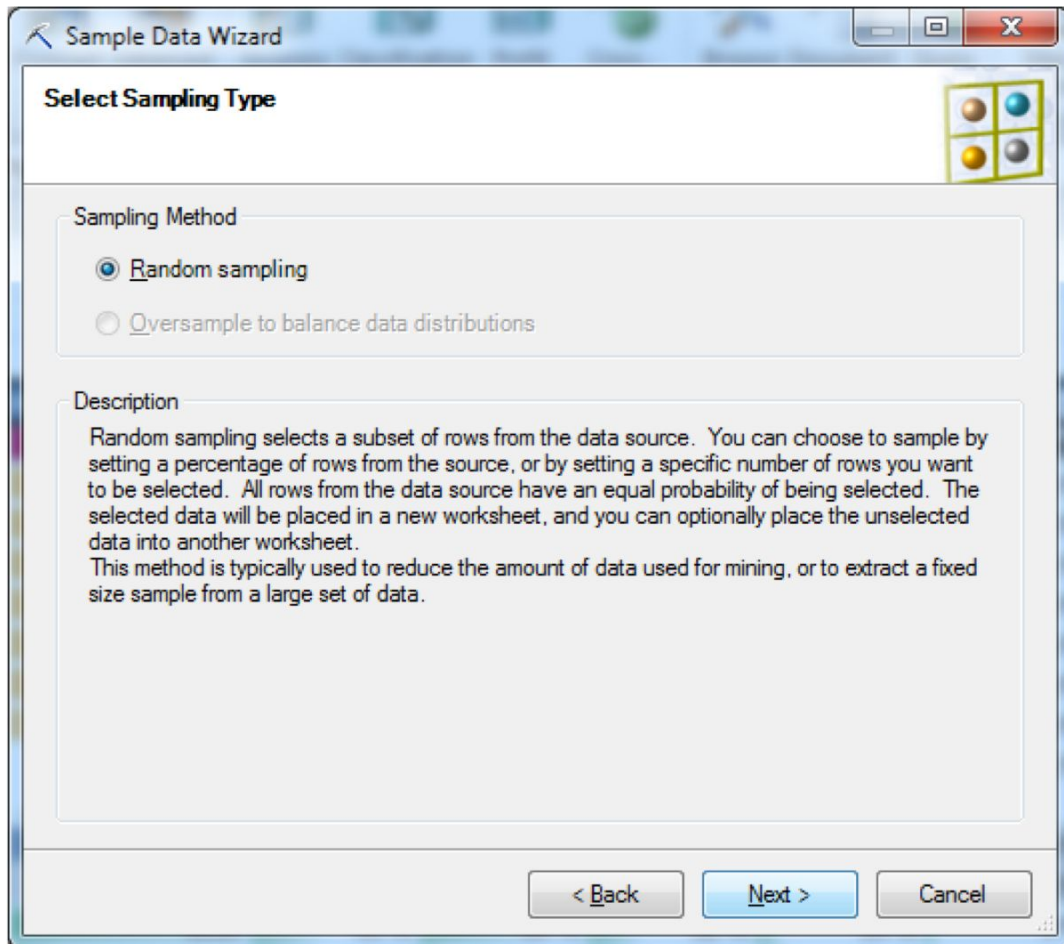


Figure 11–26. The Sample Data wizard showing random sampling as the only sampling method

Using the Data Modeling Group

The next section of the Data Mining ribbon is the Data Modeling group. This group gives you access to all of the mining model algorithms available in SSAS. Here you can implement the algorithms using Excel as source data for your mining models. You can create temporary or permanent mining models using these buttons.

Note You can create temporary mining models if, in the initial configuration of the Data Mining Add-ins for Office, you selected that option. Of course, any user who attempts to build permanent mining structures or models must have the appropriate permissions on the SSAS server.

Each button in this group is an alias for a particular SSAS data mining algorithm. If you want to create a structure or model using the mining algorithm names used in SSAS, use the Advanced button on the ribbon (see Figure 11-27). The Accuracy Chart and Classification Matrix buttons will be covered in the next section; they are part of the Accuracy and Validation section of the ribbon.

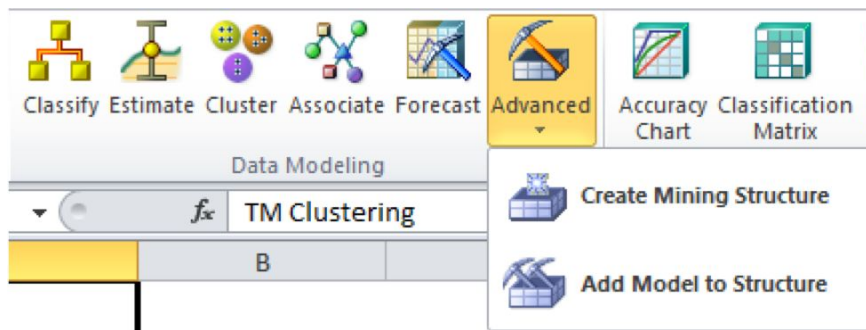


Figure 11-27. The Data Modeling section of the Data Mining ribbon

Excel has four algorithm choices using the buttons on the ribbon. You can think of these as aliases for specific data mining algorithms available in SSAS. Here's how the aliases map from Excel to SSAS:

- Excel *Classify* builds a Microsoft Decision Tree SSAS model (predicts a single value of any type).
- Excel *Estimate* builds a Microsoft Decision Tree SSAS model (predicts any continuous—numeric or datetime—value).
- Excel *Cluster* builds a Microsoft Clustering SSAS model (creates groupings of related values).
- Excel *Associate* builds a Microsoft Association Rules SSAS model (creates groupings of related items per a configured value—3 by default—or market basket).
- Excel *Forecast* builds a Microsoft Time Series SSAS model (predicts a time-based value).

As with the other buttons on the ribbon, if you click any one of these buttons, the first page of the respective wizards describes in nontechnical terms the functionality of the particular underlying algorithm. If you, or your end users, prefer to use the SSAS data mining algorithm by the original Microsoft name to create temporary or permanent mining models, then you can click the Advanced button on the ribbon, and then click Create Mining Structure or Add Model to Structure. This will launch a Mining Model wizard that is similar, but not identical, to the one available in BIDS.

For example, one difference in the Add Model to Structure dialog box is where you select the particular algorithm. In the Excel version of the dialog box, there is a new Parameters button on the bottom left; clicking it will allow you to configure the most common parameters for that particular algorithm. Figure 11–28 displays the four parameters available for the Naïve Bayes algorithm mentioned earlier in this chapter.

The SQL Server Data Mining Add-in brings the power of data mining to Excel users. Because Excel is readily available and the Data Mining Add-ins are free, the possibilities for expanding data mining's reach into a broad section of end users is well within reach of most BI solutions. The point of BI projects is to make knowledge available in a more meaningful way to more users. Excel, as a general SSAS client, facilitates this. Its deep support for data mining simply increases the reach.

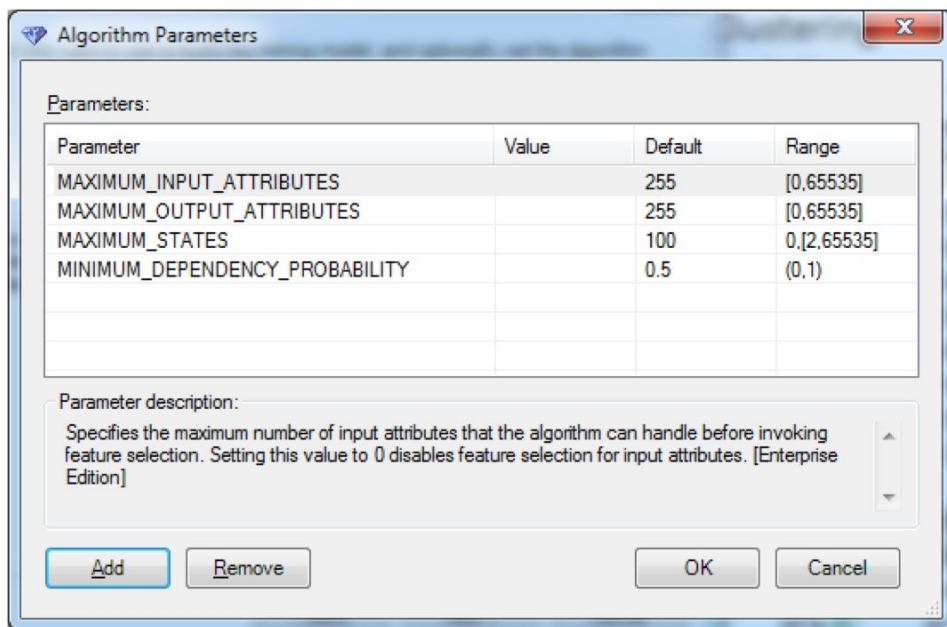


Figure 11–28. Modifying parameters for your selected algorithm

Using the Accuracy and Validation Group

After you've built your models, either in Excel or in BIDS, you will probably want to evaluate their usefulness in answering the particular business questions they've been built to address. The Accuracy and Validation group of the ribbon gives you access to the Mining Accuracy Chart functionality in BIDS, which is covered in more detail in Chapter 14. This portion of the ribbon has four buttons: Accuracy Chart, Classification Matrix, Profit Chart, and Cross-Validation (see Figure 11–29).

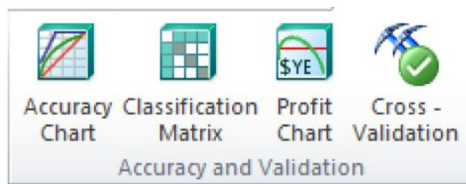


Figure 11-29. The Accuracy and Validation section of the Data Mining ribbon.

As with most of the other buttons on the Data Mining ribbon, when you click the Accuracy Chart button, a wizard opens with the first dialog box documenting in detail exactly what this wizard does (Figure 11-30). To refresh your memory, the output here is a either a lift or a profit chart—both of which are designed to visually display effectiveness of a particular mining model.

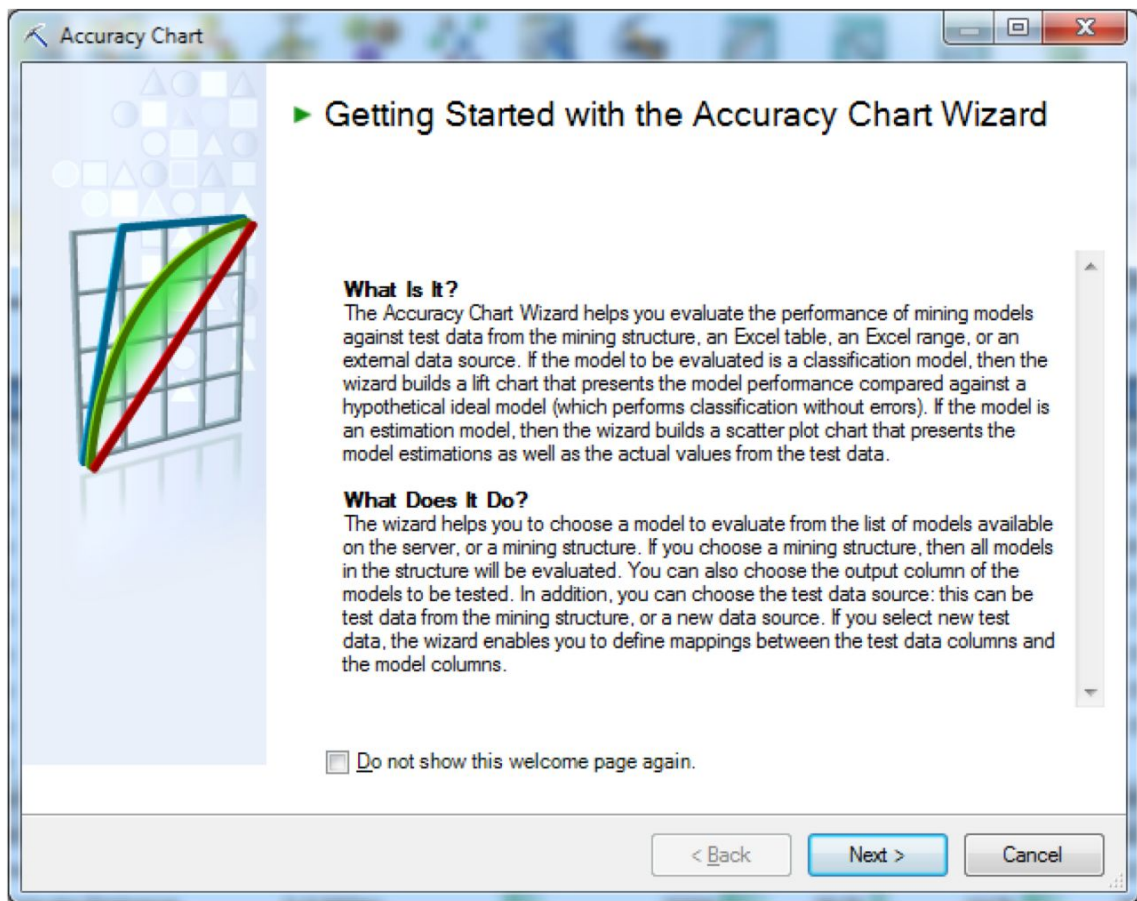


Figure 11-30. The Getting Started dialog box of the Accuracy Chart wizard

The next step is to select a model. If you attempt to validate a model that was built with an algorithm that isn't supported for this functionality, the dialog box will warn you. For this example, we are using a model built with the Microsoft Sequence Clustering algorithm, which is supported for use with an Accuracy Chart.

In the next step of the wizard, you are asked to select the mining column and value to predict. Choose a Value to predict of 1, which will create a lift chart for bike buyers (see Figure 11–31).

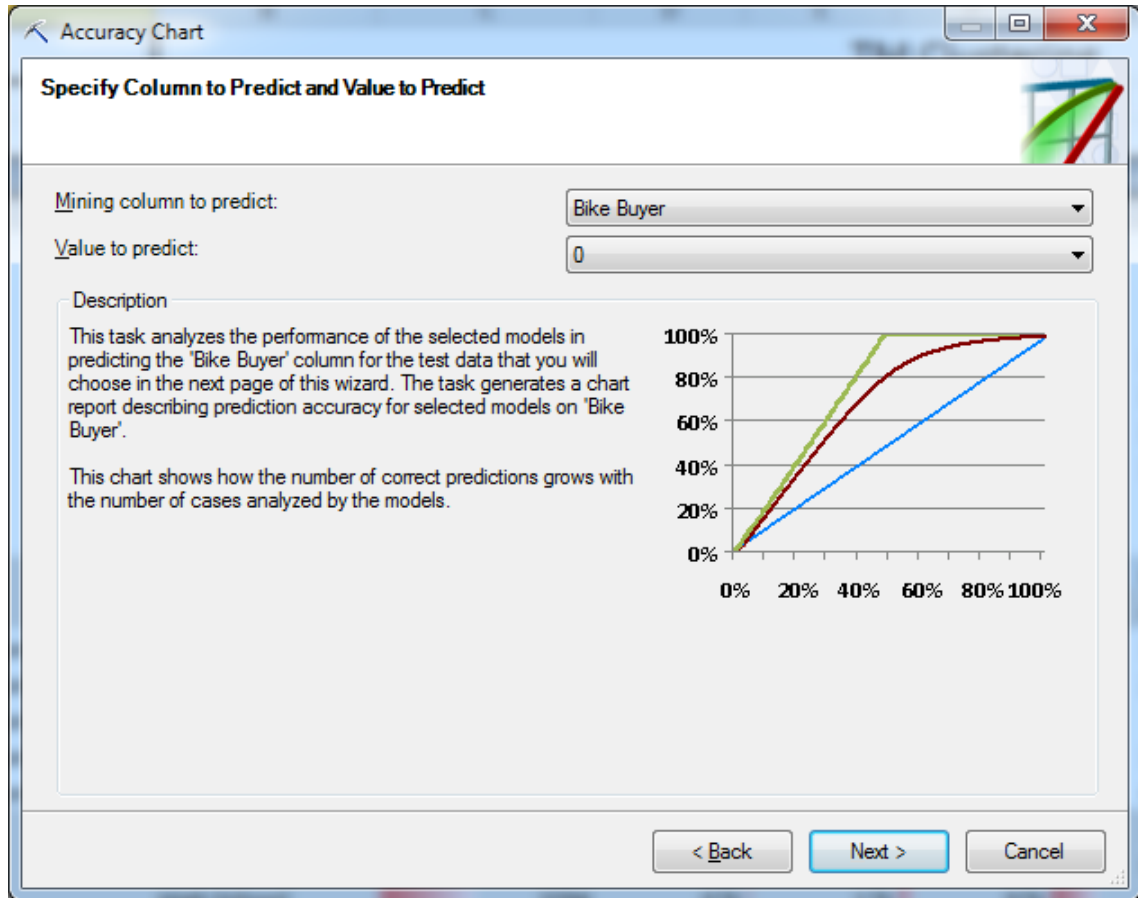


Figure 11–31. Selecting the mining column to predict and the value

Next, you select the source data, which can be a worksheet, data range, or SSAS data source. The next step asks you to specify a relationship between the mining model column(s) and the table columns. The output of the Accuracy Chart is a lift chart that is identical to using the mining Accuracy Chart functionality in BIDS. Accuracy charts and lift charts will be explored further in Chapter 14.

The Classification Matrix button lets you to see the number of correct and incorrect predictions made by your model in a spreadsheet format output. It functions identically to features found in BIDS. As with the Accuracy Chart, when you click the Classification Matrix button, the wizard opens with a very descriptive dialog box. Configuration steps are similar to those performed when using the Accuracy

Chart. The key dialog box is called Classification Matrix, in which you select the column to predict and specify options regarding the output (see Figure 11–32).

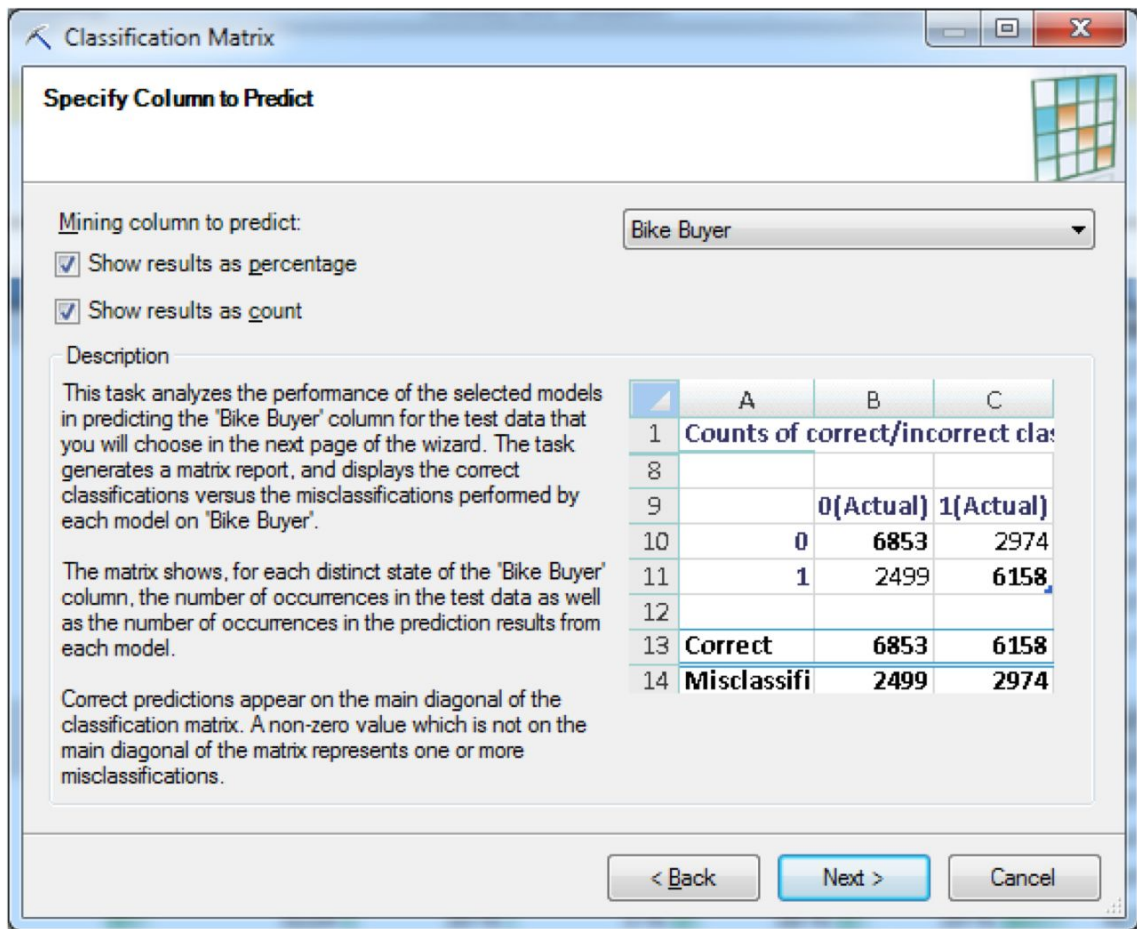
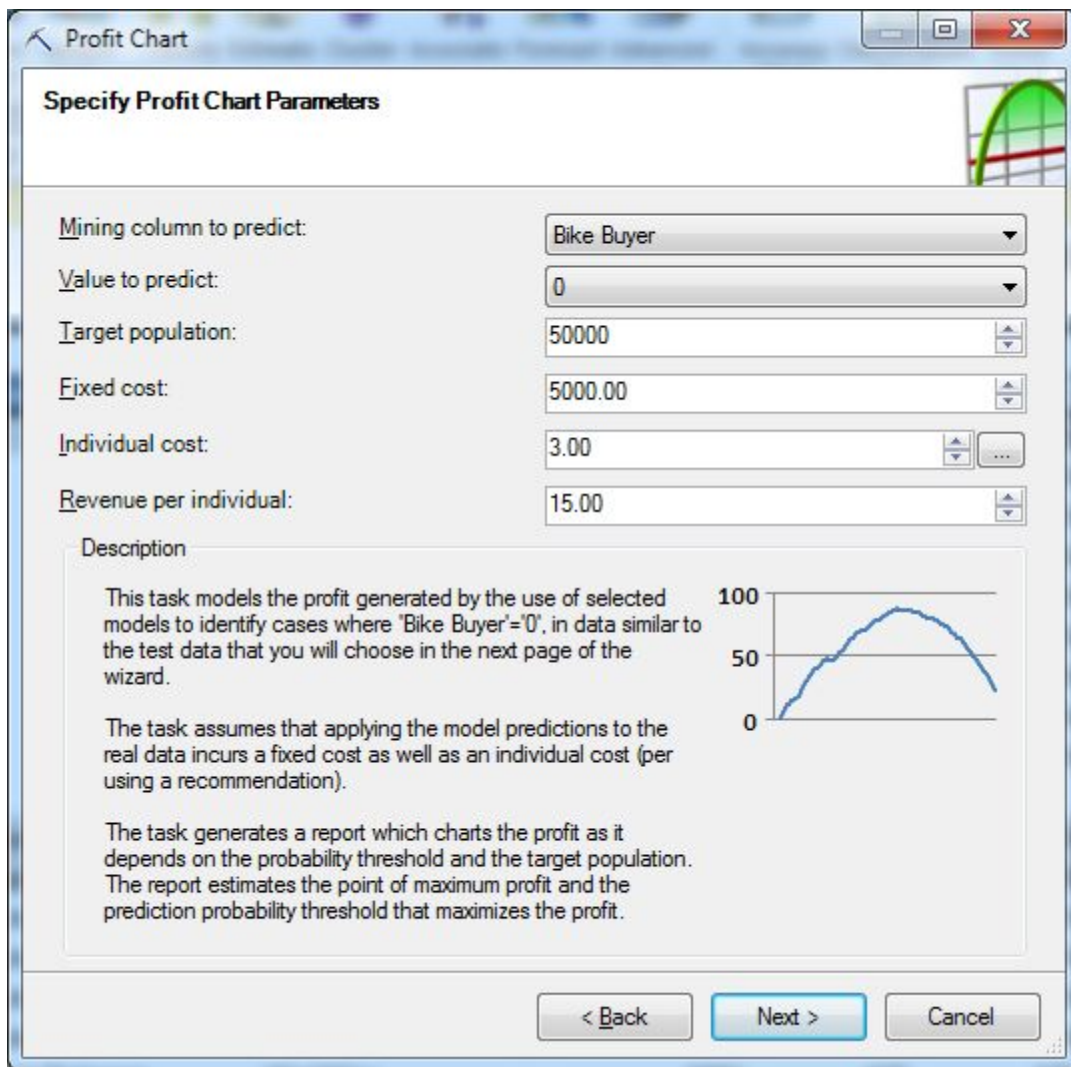


Figure 11–32. Configuring the format of the output

The Profit Chart button allows you to quickly and easily build a profit chart. The key dialog box of the wizard is the Specify Profit Chart Parameters dialog box shown in Figure 11–33.



Profit Chart

Specify Profit Chart Parameters

Mining column to predict: Bike Buyer

Value to predict: 0

Target population: 50000

Fixed cost: 5000.00

Individual cost: 3.00

Revenue per individual: 15.00

Description

This task models the profit generated by the use of selected models to identify cases where 'Bike Buyer'='0', in data similar to the test data that you will choose in the next page of the wizard.

The task assumes that applying the model predictions to the real data incurs a fixed cost as well as an individual cost (per using a recommendation).

The task generates a report which charts the profit as it depends on the probability threshold and the target population. The report estimates the point of maximum profit and the prediction probability threshold that maximizes the profit.

100
50
0

< Back Next > Cancel

Figure 11–33. Configuring the parameters needed to produce the profit chart

Summary

This chapter introduced you to data mining with Excel. We discussed configuring Excel for data mining, as well as using the Data Preparation Group and the Data Modeling Group of tools. In the next chapter, you will discover PowerPivot, Microsoft's next-generation end-user OLAP tool.



Introducing PowerPivot

PowerPivot is an exciting add-in to Microsoft Excel that allows your users to do their own data analysis and mining. Using PowerPivot, you can connect to a database, bring large amounts of data into Excel, perform analysis upon that data, and push the results back up to your server for others in your organization to view. PowerPivot's extraordinary functionality gives you full control over your data analysis and mining efforts.

The PowerPivot for Excel GUI

PowerPivot allows you to use Excel as a central workspace for all of your analytics. A PowerPivot workbook can connect to a SQL Server database and use data from Excel spreadsheets. Using PowerPivot, you can enrich your data by creating custom calculations and by defining data relationships. PowerPivot stores data in your workbook directly, and these workbooks can contain millions of rows of data.

The Excel add-in for PowerPivot is a free download from www.microsoft.com. The best way to find it is to start at www.powerpivot.com/download.aspx, which has links to both the 32-bit and 64-bit versions of the Excel add-in. Note that the 32-bit or 64-bit versions are with respect to what version of Excel you installed, not your operating system. If you have 32-bit (x86) Excel running on 64-bit (x64) Windows 7, use the 32-bit add-in.

■ **Note** To take full advantage of PowerPivot for Excel, you need Excel 2010. Excel 2010 is the first version of Excel to allow add-ins to store custom data within a workbook. Excel 2007 can be used to open PowerPivot workbooks.

The PowerPivot Ribbon

The Excel client adds a tab to the ribbon for PowerPivot, as shown in Figure 12-1. This is used for managing the PowerPivot client. Think of PowerPivot as a cube running under Excel—the PowerPivot tab gives you access to the designer, which allows you to create calculated measures from the underlying data, add pivot tables and charts based on the PowerPivot cube to the workbook, and link in data from tables in the workbook.

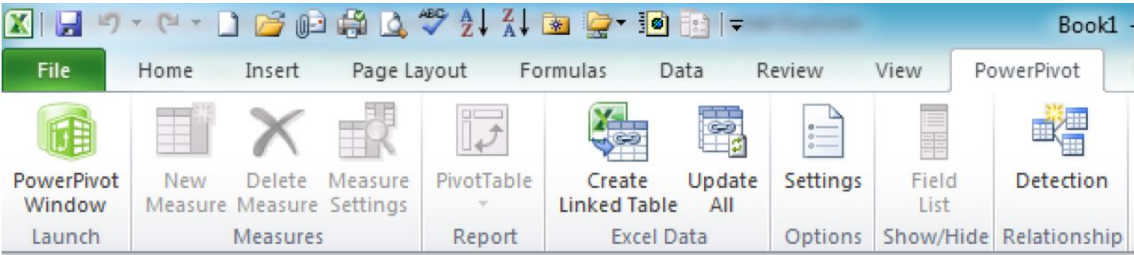


Figure 12-1. PowerPivot tab in Excel 2010

Clicking the PowerPivot Window button opens the PowerPivot designer, which is like a stripped-down version of Excel focused on data. Figure 12-2 shows the PowerPivot Designer. Along the bottom you will see multiple tables of data. You can import data from any number of sources. Out of the box, PowerPivot supports connections to multiple data sources including SQL Server, SQL Azure, Access, Oracle, Teradata, Excel, Text Files, and Reporting Services.

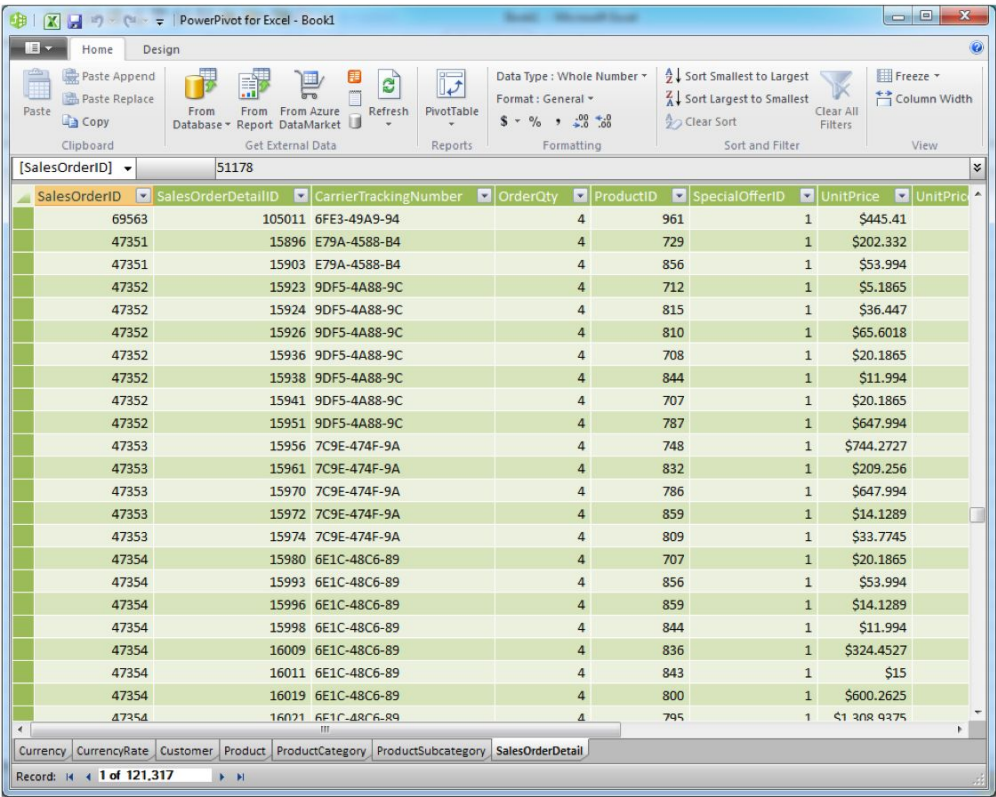


Figure 12-2. The PowerPivot Designer

The Measures group contains three items that allow you to manage your PowerPivot measures: New Measure allows you to create new measures, Delete Measure will delete a measure, and you can make any edits to your custom measures via the Measure Settings dialog. Figure 12–3 shows the Measure Settings dialog.

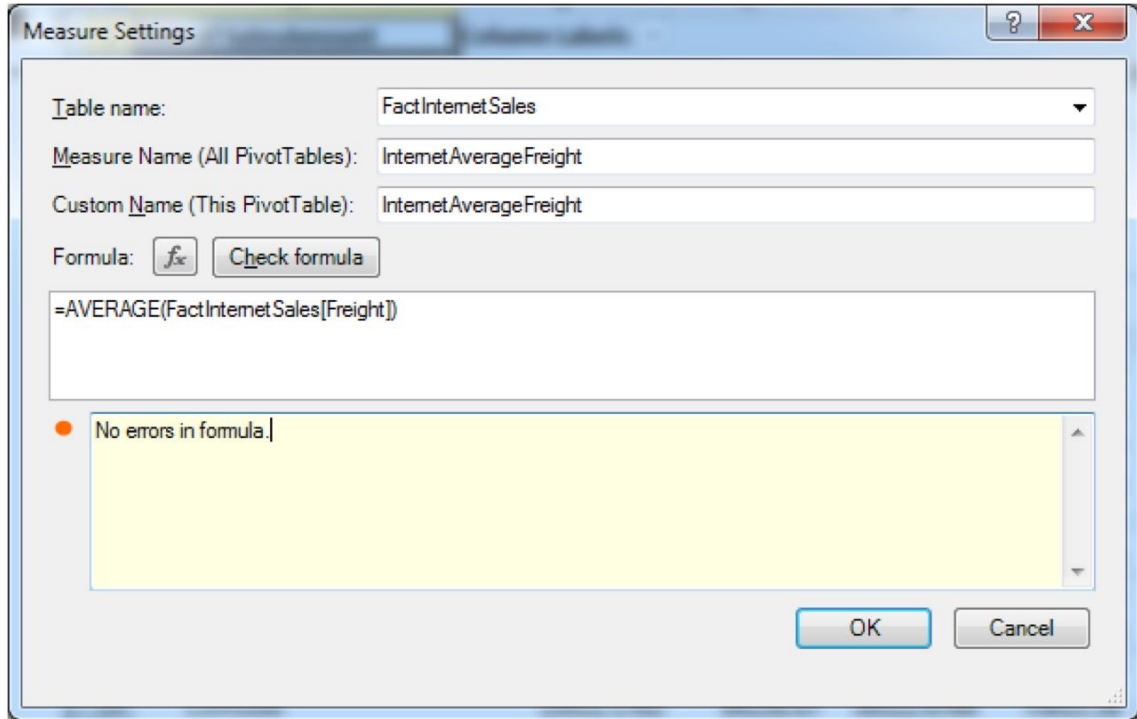


Figure 12–3. The Measure Settings dialog for creating and editing measures

The Report group lists the report types available to your workbook. There are a number of report types available (see Figure 12–4), including PivotTable and PivotChart.

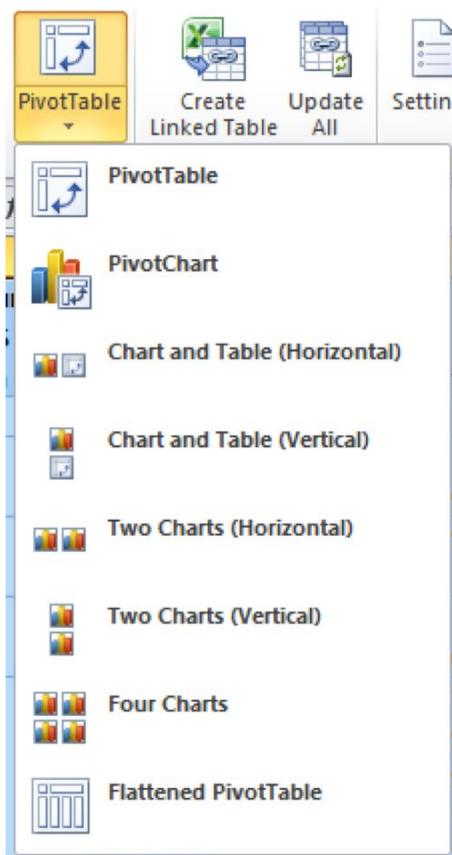


Figure 12–4. The report types available in the Report group

The Excel Data group is where you begin to work with Excel data in PowerPivot. The Create Linked Table selection allows you to link data created in Excel to a table in PowerPivot. Linking Excel data in this way allows you to continue working with your Excel data while analyzing any changes in PowerPivot.

The Options group has a single selection, Settings, that will display the PowerPivot Options & Diagnostics dialog. This is where you can find your PowerPivot version number.

The Show/Hide group contains the Field List selection. When selected, the PowerPivot Field List pane will be visible.

Finally, the Relationship group contains the Detection selection. When selected, PowerPivot will automatically attempt to find relationships in your data.

The PowerPivot Designer

The PowerPivot designer is the main PowerPivot work area. In this window, there are two tabs in the designer ribbon: the Home tab and the Design tab. The Home tab is where you will add and format your data, while the Design tab allows you to create relationships and add columns.

The Home tab contains the following six groups: Clipboard, Get External Data, Reports, Formatting, Sort and Filter, and View. Figure 12–5 displays the Home tab.

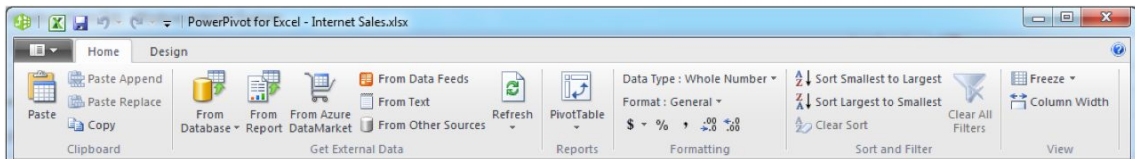


Figure 12–5. The Home tab

The Clipboard group is where you will perform standard copy and paste functions. One useful feature of the Clipboard group is that it allows you to append or replace the data you are pasting.

The Get External Data group allows you to get data into PowerPivot from a variety of sources. The available sources include databases, Excel reports, and text files. Clicking Refresh will update your external data.

The Reports group allows you to insert any of the available report types into an Excel workbook using your PowerPivot data, while the Formatting group contains standard formatting items.

The Sort and Filter group is where you can sort columns in ascending or descending order. The Clear All Filters button will clear any filters set in your data. Note that setting filters in your data works the same way as using AutoFilter in Excel.

The View group is where you can set column widths and freeze or unfreeze columns.

The Design tab contains the following groups: Columns, Calculations, Connections, Relationships, Properties, and Edit. Figure 12–6 displays the Design tab.

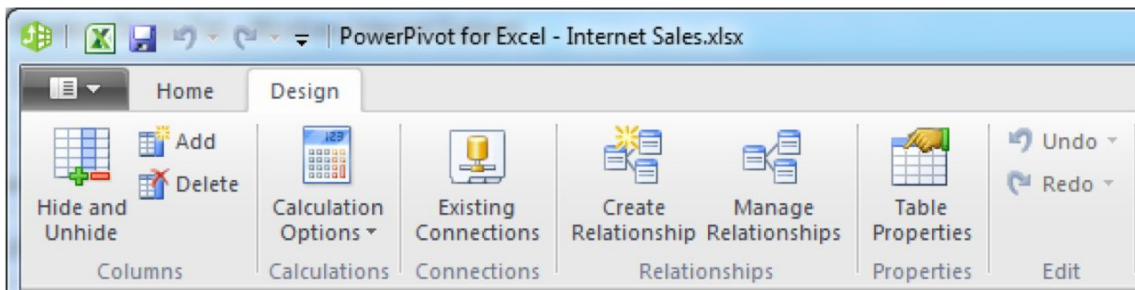


Figure 12–6. The Design tab

The Columns group allows you to add and delete columns. You can also use this group to hide columns.

The Calculations group is where you set your preference for recalculating your PowerPivot data. You can set this option to Manual Calculation Mode, if you would like to control when recalculation occurs, or Automatic Calculation Mode, which will recalculate your data whenever a change causes a need for recalculation.

The Connections group is where you edit existing data connections. You may, for example, want to use a development or testing database. Figure 12–7 shows the Edit Connection dialog for an existing PowerPivot data connection.

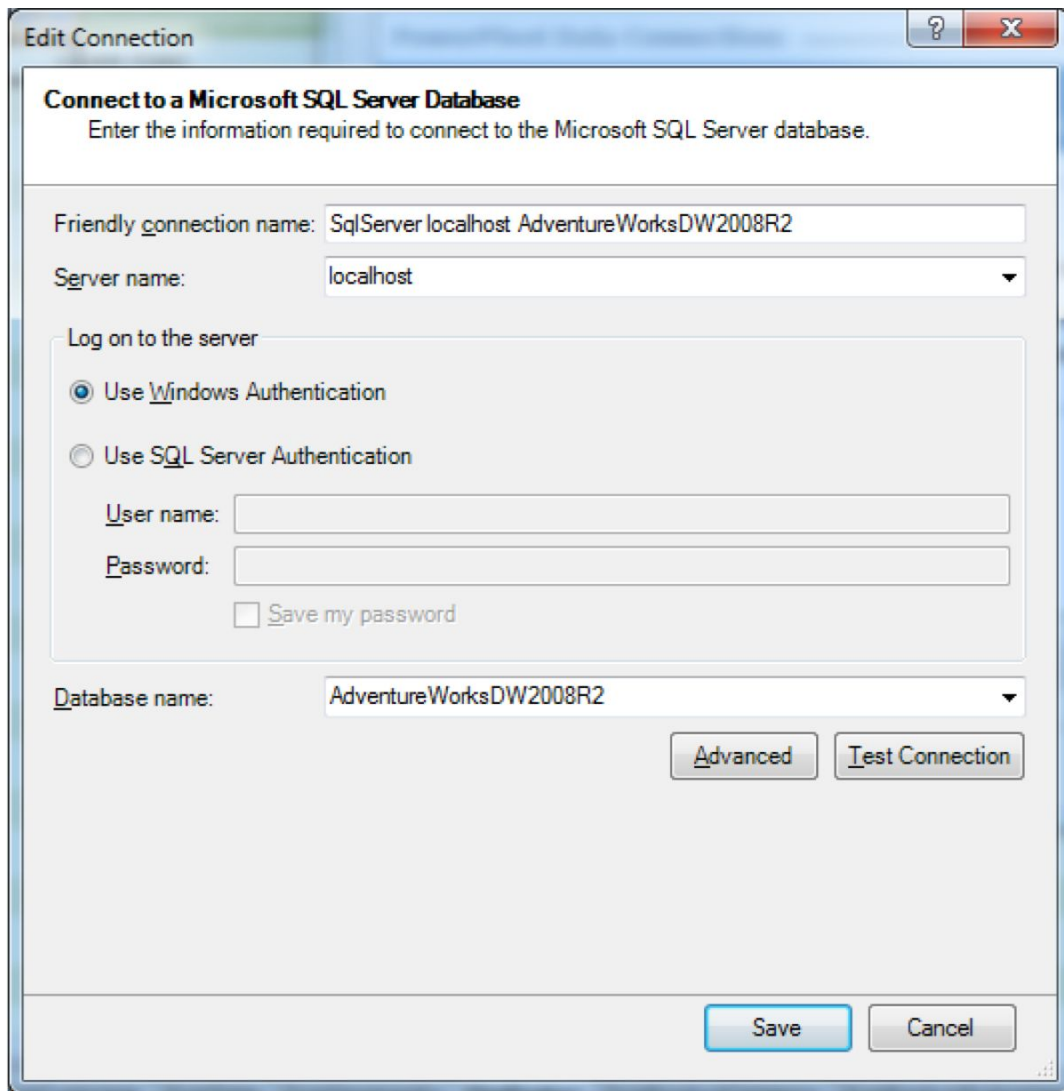


Figure 12–7. The Edit Connection dialog

The Relationships group contains two buttons: Create Relationship and Manage Relationships. Create Relationship allows you to create standard one-to-one and one-to-many relationships in your PowerPivot data. Figure 12–8 shows an example of a one-to-many relationship between DimProductSubcategory and DimProductCategory.

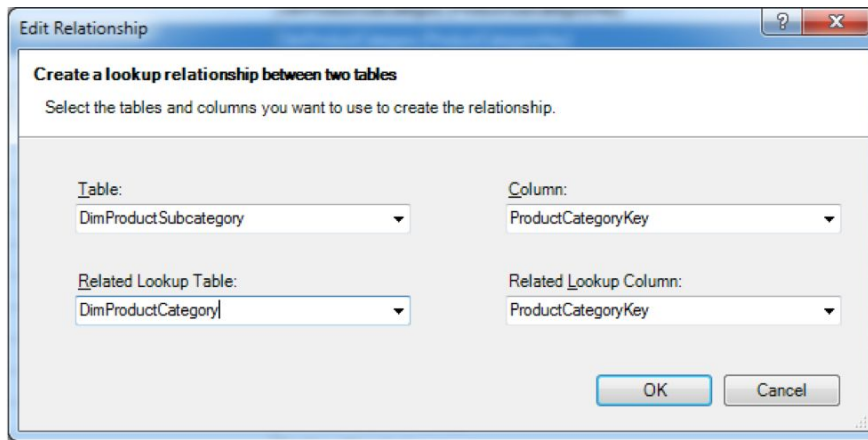


Figure 12-8. Viewing a PowerPivot data relationship

The Properties group has a single button, Table Properties, where you to view and edit table properties, including columns and filters. The Table Properties dialog also enables editing of the source SQL used to import your PowerPivot data. Figure 12-9 shows the Edit Table Properties dialog.

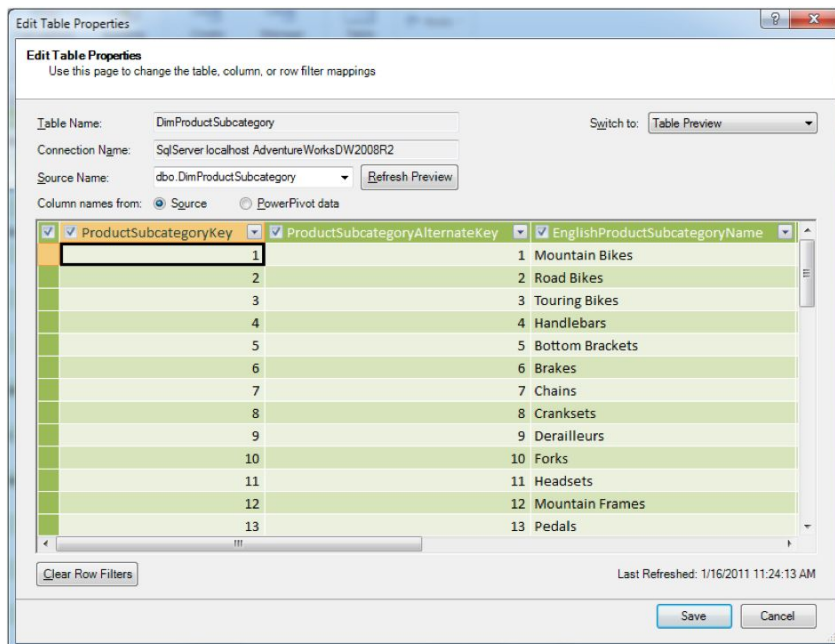


Figure 12-9. The Edit Table Properties dialog

Finally, the Edit group contains standard Undo and Redo buttons.

Using PowerPivot with Adventure Works

In this section, you will explore Adventure Works using PowerPivot. You will learn how to import data into PowerPivot and how to enrich your PowerPivot data with additional columns and calculations. After finalizing your data set, you will post your PowerPivot data into Excel.

Importing Adventure Works Data

To set the stage for using PowerPivot, you must first open Excel. When Excel is loaded, select the PowerPivot tab, and click PowerPivot Window. From the empty PowerPivot designer, click From Database, and select From SQL Server. In the Table Import wizard, enter a connection name, server name, and select AdventureWorksDW2008R2 as your SQL Server database. The completed dialog is shown in Figure 12–10.

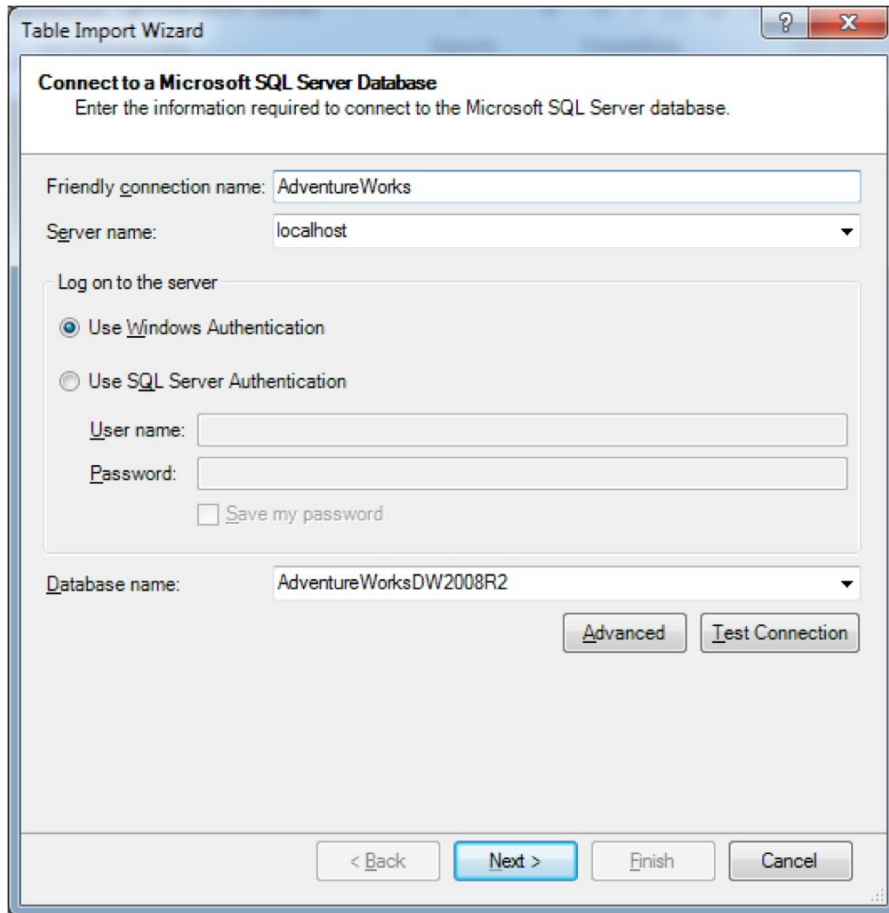


Figure 12–10. The Connect to a Microsoft SQL Server Database page of the Table Import Wizard dialog

Once you have completed your entries, test the connection using the Test Connection button, then click Next to continue. In the Choose How to Import the Data dialog, choose “Select from a list of tables and views to choose the data to import,” and click Next.

This will display the Select Tables and Views dialog. In this dialog, select the DimCustomer and DimProductSubcategory tables. Next, click Select Related Tables. PowerPivot will select tables and views in Adventure Works that are related to DimCustomer and DimProductSubcategory; it will mark five tables as related. Next, select DimDate as a source table. Figure 12–11 shows the Select Tables and Views dialog with eight tables selected.

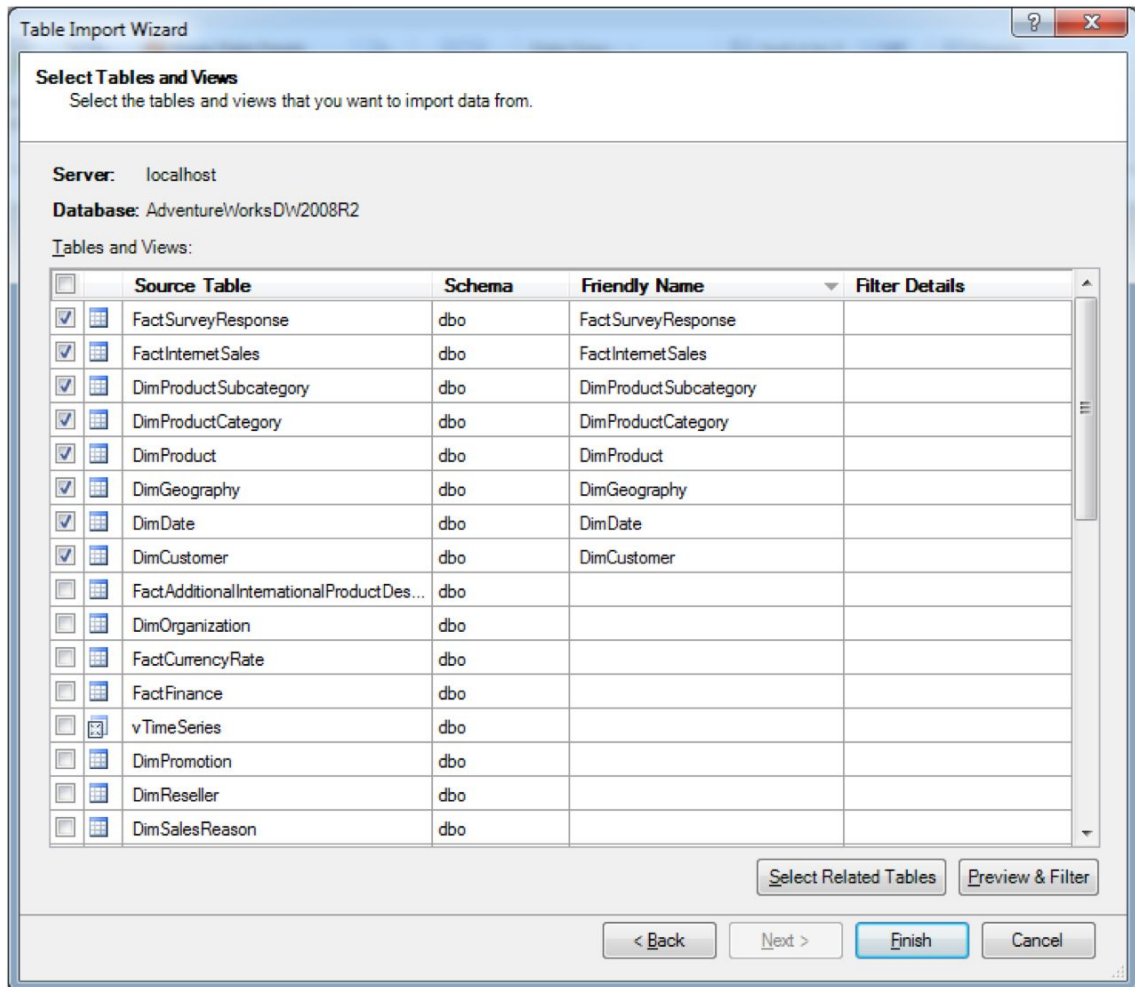


Figure 12–11. The Select Tables and Views dialog with selected tables at the top

Click Finish to complete the wizard. PowerPivot will now import your data. When the import finishes, click Close to dismiss the dialog, and return to PowerPivot. Figure 12–12 displays the PowerPivot designer with eight tabs along the bottom, one for each table imported.

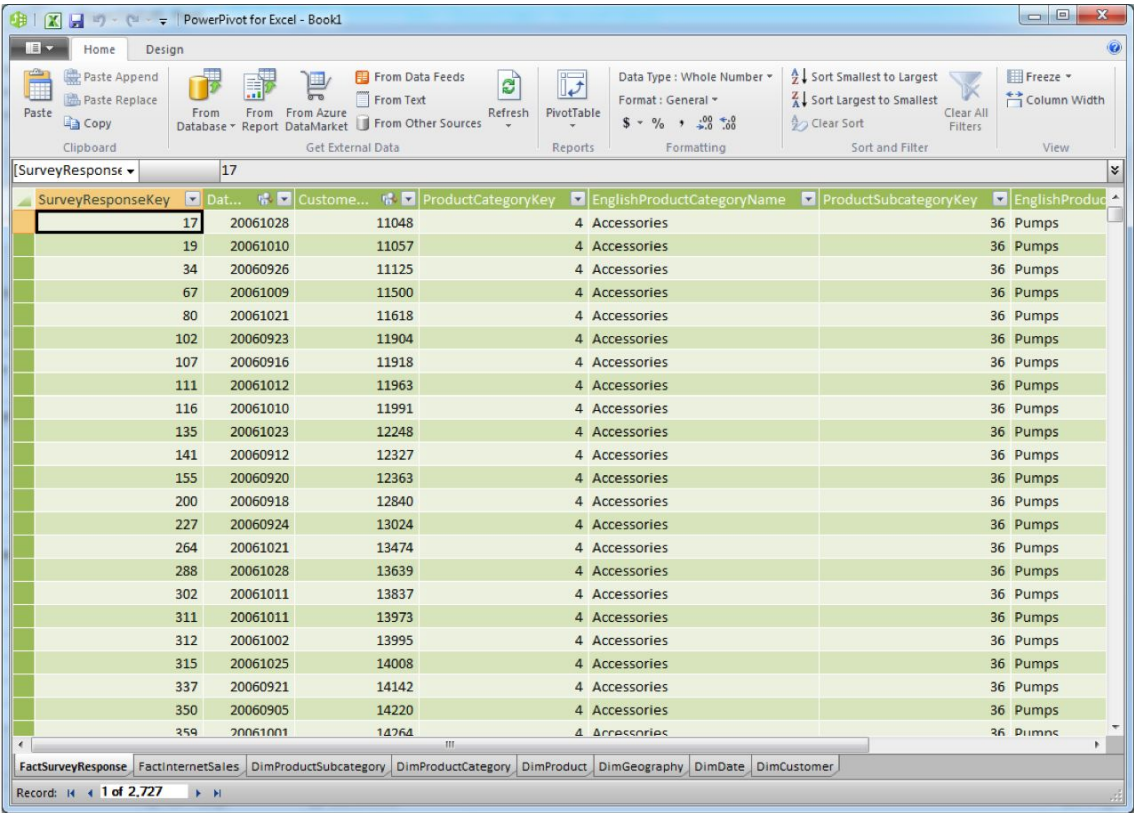


Figure 12–12. The PowerPivot designer after importing eight tables from Adventure Works

Enriching the Adventure Works Data

As you work with and analyze your data, you will probably need to remove uninteresting data points, filter data to enhance your analysis, and perform custom calculations. These analysis activities are referred to as *enriching* your data. In the following sections, you will first learn how to filter and sort your PowerPivot data. You will also hide meaningless data and create a calculated column.

Sorting and Filtering

Let’s start by sorting the FactInternetSales table by OrderDateKey in descending order. To begin, select the FactInternetSales tab, and then select the OrderDateKey column header. Clicking the drop-down in the OrderDateKey column header will display the sorting and filter options for this column, as shown in Figure 12–13.

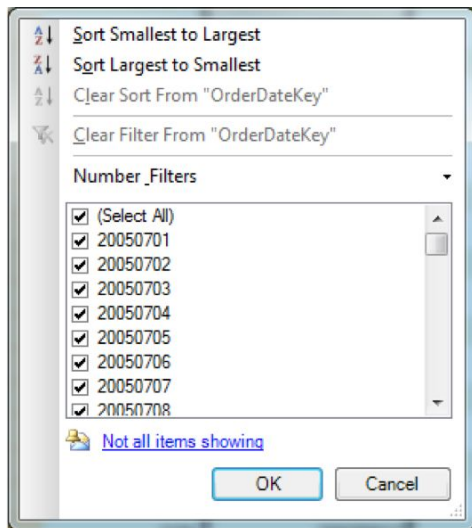


Figure 12-13. The sort and filter dialog for the OrderDateKey column

Clicking on Sort Largest to Smallest will sort the OrderDateKey column in descending order. PowerPivot offers several options for filtering your data, using the same dialog shown in Figure 12-13. Clicking on the drop-down to the right of Number_Filters opens the menu shown in Figure 12-14.

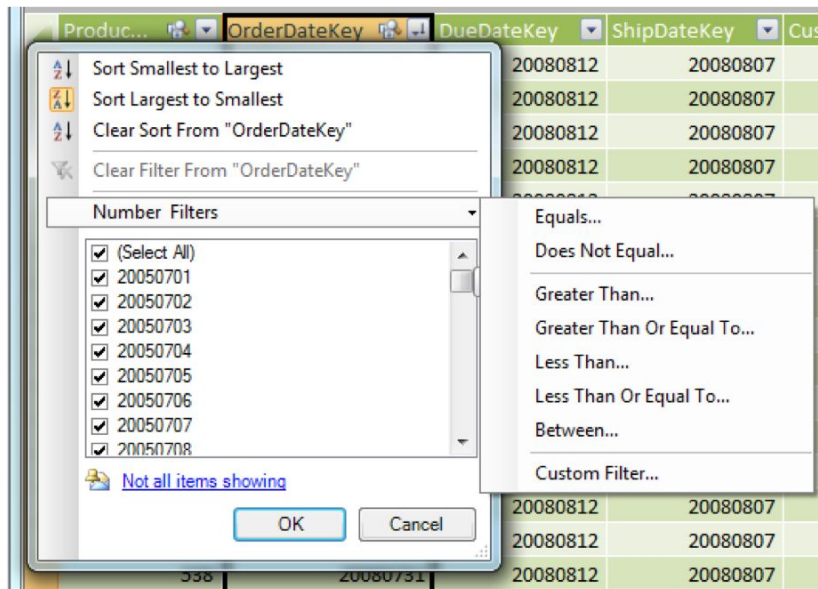


Figure 12-14. The sort and filter dialog with the filter menu shown for the OrderDateKey column

Using the Greater Than Or Equal To menu item, filter the OrderDateKey to contain only sales that occurred since 2008 by entering 20080101.

■ **Note** The various date keys in Adventure Works are integer representations of the International Date Format (yyyy-mm-dd). Keys that give information about the underlying data are often referred to as *smart keys*.

After you have completed these steps, take note of the icons in the OrderDateKey column header. The rightmost icon shows that the OrderDateKey column is sorted and filtered. If you hover over the left icon, PowerPivot will display the relationships in which the OrderDateKey column participates.

Hiding Columns

To narrow the width of FactInternetSales and prepare for adding calculated columns, choose the Design tab, and click Hide and Unhide. This will load the Hide and Unhide Columns dialog. Next, deselect the following columns: PromotionKey, CurrencyKey, SalesTerritoryKey, SalesOrderNumber, SalesOrderLineNumber, RevisionNumber, UnitPriceDiscountPct, DiscountAmount, ProductStandardCost, SalesAmount, TaxAmt, Freight, CarrierTrackingNumber, and CustomerPONumber. Your Hide and Unhide Columns dialog should look like Figure 12–15.

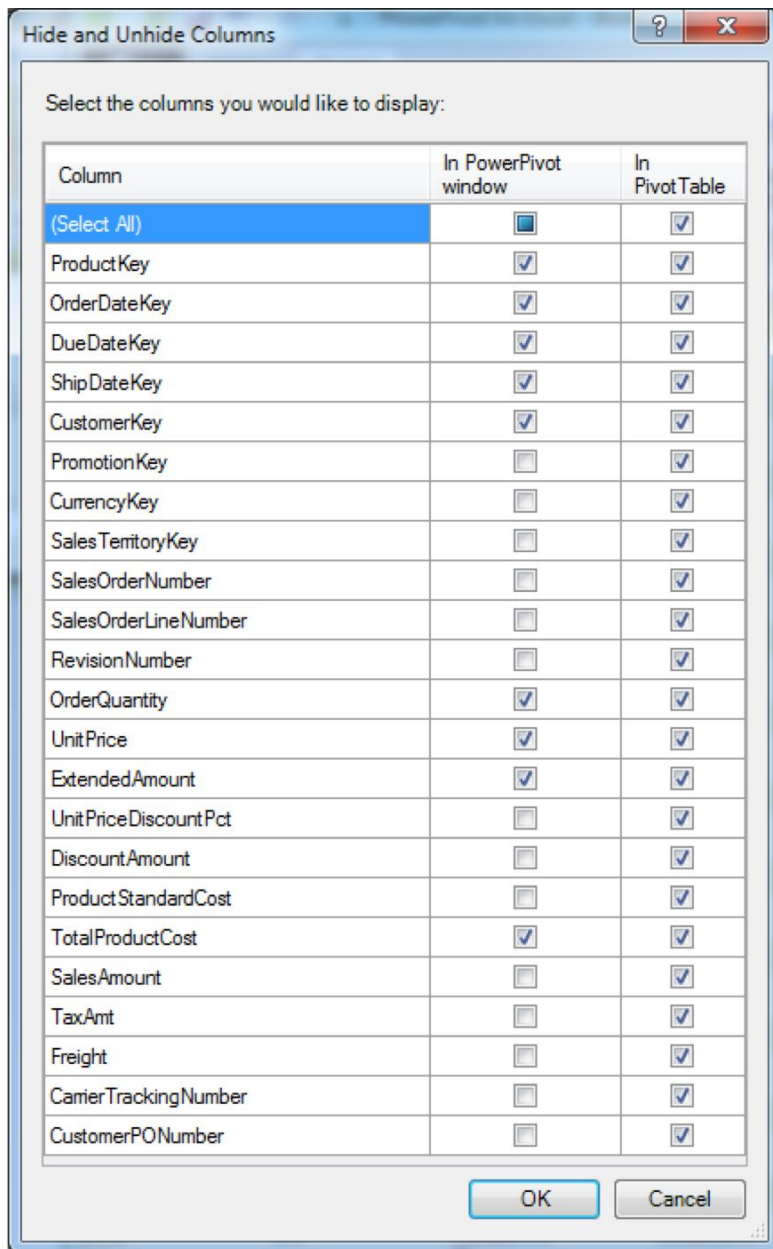
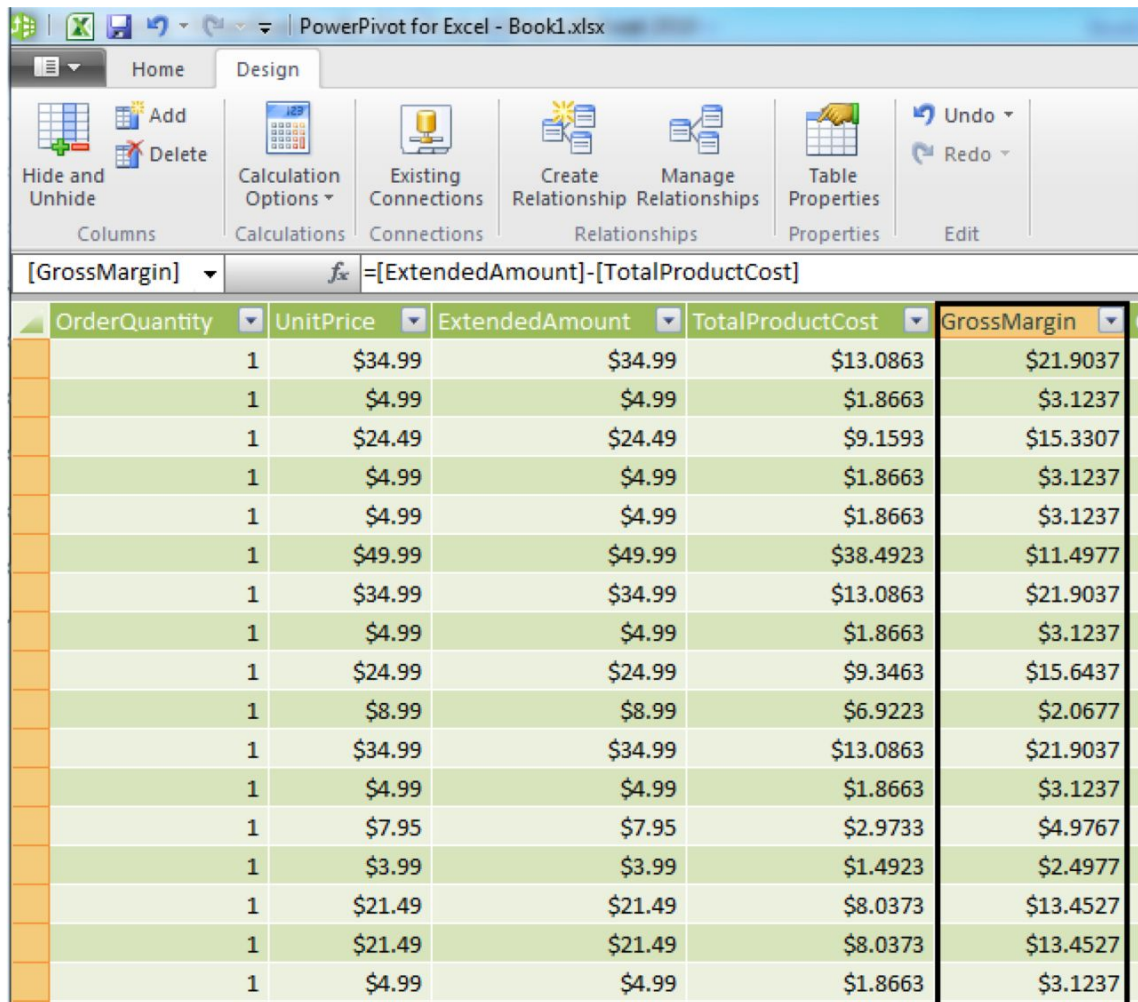


Figure 12–15. The Hide and Unhide Columns dialog with several columns deselected

Adding Calculated Columns

Creating a calculated column in PowerPivot uses Excel-like formulas that apply to an entire column. PowerPivot uses a new language to create formulas, called Data Analysis Expressions (DAX). PowerPivot DAX is a powerful extension to the Excel formula set. A full DAX reference is available on MSDN at <http://msdn.microsoft.com/en-es/library/ee634556.aspx>.

To create a calculated column, click the Add button in the Columns group to add a new column to the end of FactInternetSales. Double-click the Add Column column header, and enter GrossMargin for a column name. Define this column by entering `= [ExtendedAmount] - [TotalProductCost]` in the formula bar, and press Enter. PowerPivot calculates every row and then displays the results, as shown in Figure 12–16.



| | OrderQuantity | UnitPrice | ExtendedAmount | TotalProductCost | GrossMargin |
|--|---------------|-----------|----------------|------------------|-------------|
| | 1 | \$34.99 | \$34.99 | \$13.0863 | \$21.9037 |
| | 1 | \$4.99 | \$4.99 | \$1.8663 | \$3.1237 |
| | 1 | \$24.49 | \$24.49 | \$9.1593 | \$15.3307 |
| | 1 | \$4.99 | \$4.99 | \$1.8663 | \$3.1237 |
| | 1 | \$4.99 | \$4.99 | \$1.8663 | \$3.1237 |
| | 1 | \$49.99 | \$49.99 | \$38.4923 | \$11.4977 |
| | 1 | \$34.99 | \$34.99 | \$13.0863 | \$21.9037 |
| | 1 | \$4.99 | \$4.99 | \$1.8663 | \$3.1237 |
| | 1 | \$24.99 | \$24.99 | \$9.3463 | \$15.6437 |
| | 1 | \$8.99 | \$8.99 | \$6.9223 | \$2.0677 |
| | 1 | \$34.99 | \$34.99 | \$13.0863 | \$21.9037 |
| | 1 | \$4.99 | \$4.99 | \$1.8663 | \$3.1237 |
| | 1 | \$7.95 | \$7.95 | \$2.9733 | \$4.9767 |
| | 1 | \$3.99 | \$3.99 | \$1.4923 | \$2.4977 |
| | 1 | \$21.49 | \$21.49 | \$8.0373 | \$13.4527 |
| | 1 | \$21.49 | \$21.49 | \$8.0373 | \$13.4527 |
| | 1 | \$4.99 | \$4.99 | \$1.8663 | \$3.1237 |

Figure 12–16. The GrossMargin calculated column with its formula

You probably noticed, by looking at the formula, that PowerPivot does not have the concept of a “cell.” In this regard, PowerPivot operates more like a database, using column names to perform operations.

In this section, you will add a cycle-time metric to FactInternetSales. This metric will measure the number of days needed to fulfill an order from order date to ship date. To begin, click the Add button in the Columns group to add a new column to the end of FactInternetSales. Double-click the Add Column column header and enter OrderShipCycleDays for a column name.

Next, add two more columns using the same procedure, and name them OrderDate and ShipDate. You will create OrderDate and ShipDate using two different methods: basic formatting and related data.

To convert the ship date key, which is an integer, into an actual date, you will use the Date function. The Date function takes three integer inputs: year, month, and day. You will create the three needed parameters by parsing the ShipDateKey into the three parts needed using the Left, Mid, and Right functions. In the formula bar, enter the following code for the ShipDate column:

```
=DATE(Left(FactInternetSales[ShipDateKey],4),
Mid(FactInternetSales[ShipDateKey],5,2),
Right(FactInternetSales[ShipDateKey],2))
```

This formula and its results are shown in Figure 12–17.

| =DATE(Left(FactInternetSales[ShipDateKey],4), Mid(FactInternetSales[ShipDateKey],5,2), Right(FactInternetSales[ShipDateKey],2)) | | | | | | | | | |
|---|---------------|-----------|----------------|------------------|-------------|--------------------|-----------|----------|--|
| ShipDateKey | OrderQuantity | UnitPrice | ExtendedAmount | TotalProductCost | GrossMargin | OrderShipCycleDays | OrderDate | ShipDate | |
| 18529 | 1 | \$34.99 | \$34.99 | \$13.0863 | \$21.9037 | | | 8/7/2008 | |
| 18529 | 1 | \$4.99 | \$4.99 | \$1.8663 | \$3.1237 | | | 8/7/2008 | |
| 18178 | 1 | \$24.49 | \$24.49 | \$9.1593 | \$15.3307 | | | 8/7/2008 | |
| 18178 | 1 | \$4.99 | \$4.99 | \$1.8663 | \$3.1237 | | | 8/7/2008 | |
| 16402 | 1 | \$4.99 | \$4.99 | \$1.8663 | \$3.1237 | | | 8/7/2008 | |
| 21524 | 1 | \$49.99 | \$49.99 | \$38.4923 | \$11.4977 | | | 8/7/2008 | |
| 21524 | 1 | \$34.99 | \$34.99 | \$13.0863 | \$21.9037 | | | 8/7/2008 | |
| 21524 | 1 | \$4.99 | \$4.99 | \$1.8663 | \$3.1237 | | | 8/7/2008 | |
| 21524 | 1 | \$24.99 | \$24.99 | \$9.3463 | \$15.6437 | | | 8/7/2008 | |
| 19072 | 1 | \$8.99 | \$8.99 | \$6.9223 | \$2.0677 | | | 8/7/2008 | |

Figure 12–17. The ShipDate calculated column with its formula

The Related function in PowerPivot works similarly to a database inner join. In PowerPivot, the Related function accepts a fully qualified column name as its only argument. The trick to using Related is that the parameter you pass will be the column you want to return from the *related* table.

In the case of OrderDate, PowerPivot maintains the relationship between FactInternetSales and DimDate. FullDateAlternateKey is date field you want to use from DimDate, and it can be retrieved using the following formula:

```
=RELATED('DimDate'[FullDateAlternateKey])
```

This formula and its results are shown in Figure 12–18.

=RELATED('DimDate'[FullDateAlternateKey])

| OrderKey | OrderQuantity | UnitPrice | ExtendedAmount | TotalProductCost | GrossMargin | OrderShipCycleDays | OrderDate | ShipDate |
|----------|---------------|-----------|----------------|------------------|-------------|--------------------|-----------|----------|
| 18529 | 1 | \$34.99 | \$34.99 | \$13.0863 | \$21.9037 | | 7/31/2008 | 8/7/2008 |
| 18529 | 1 | \$4.99 | \$4.99 | \$1.8663 | \$3.1237 | | 7/31/2008 | 8/7/2008 |
| 18178 | 1 | \$24.49 | \$24.49 | \$9.1593 | \$15.3307 | | 7/31/2008 | 8/7/2008 |
| 18178 | 1 | \$4.99 | \$4.99 | \$1.8663 | \$3.1237 | | 7/31/2008 | 8/7/2008 |
| 16402 | 1 | \$4.99 | \$4.99 | \$1.8663 | \$3.1237 | | 7/31/2008 | 8/7/2008 |
| 21524 | 1 | \$49.99 | \$49.99 | \$38.4923 | \$11.4977 | | 7/31/2008 | 8/7/2008 |
| 21524 | 1 | \$34.99 | \$34.99 | \$13.0863 | \$21.9037 | | 7/31/2008 | 8/7/2008 |
| 21524 | 1 | \$4.99 | \$4.99 | \$1.8663 | \$3.1237 | | 7/31/2008 | 8/7/2008 |

Figure 12–18. The OrderDate calculated column with its formula

Another function type in PowerPivot is a function that returns a table. To create OrderShipCycleDays, you will use the DatesBetween and CountRows functions. DatesBetween accepts three parameters and returns a single-column table as a result. The CountRows function works identically to Count in SQL, returning the number of rows in a resultset. You will use DatesBetween to create a single-column table containing FullDateAlternateKey values from DateDim. This is accomplished using the following code, which passes OrderDate and ShipDate to DatesBetween:

DATESBETWEEN(DimDate[FullDateAlternateKey], [OrderDate], [ShipDate])

Once DatesBetween returns its resultset, you simply pass that resultset as a parameter to CountRows, as in the following line of code:

=COUNTROWS(DATESBETWEEN(DimDate[FullDateAlternateKey], [OrderDate], [ShipDate]))

This formula and its results are shown in Figure 12–19.

=COUNTROWS(DATESBETWEEN(DimDate[FullDateAlternateKey], [OrderDate], [ShipDate]))

| TotalProductCost | GrossMargin | OrderShipCycleDays | OrderDate | ShipDate |
|------------------|-------------|--------------------|-----------|-----------|
| \$11.2163 | \$18.7737 | 8 | 7/7/2008 | 7/14/2008 |
| \$11.2163 | \$18.7737 | 8 | 7/7/2008 | 7/14/2008 |
| \$38.4923 | \$11.4977 | 8 | 7/7/2008 | 7/14/2008 |
| \$8.2205 | \$13.7595 | 8 | 7/7/2008 | 7/14/2008 |
| \$9.1593 | \$15.3307 | 8 | 7/7/2008 | 7/14/2008 |
| \$3.7363 | \$6.2537 | 8 | 7/7/2008 | 7/14/2008 |
| \$1.8663 | \$3.1237 | 8 | 7/7/2008 | 7/14/2008 |
| \$13.0863 | \$21.9037 | 8 | 7/7/2008 | 7/14/2008 |
| \$0.8565 | \$1.4335 | 8 | 7/7/2008 | 7/14/2008 |
| \$1.8663 | \$3.1237 | 8 | 7/7/2008 | 7/14/2008 |
| \$23.749 | \$39.751 | 8 | 7/7/2008 | 7/14/2008 |

Figure 12–19. The OrderShipCycleDays calculated column with its formula

This finishes your exploration of Adventure Works data using the PowerPivot designer. Now that you've learned some PowerPivot functionality and enriched the Adventure Works data, let's move back to Excel.

Using PowerPivot Data in Excel

In this section, you will move your enriched PowerPivot data into Excel. Using Excel, you will then add a slicer to your worksheet. *Slicers* allow you to dynamically filter your PivotTables within Excel. First, however, you will need to get your PowerPivot data into Excel.

To create an Excel PivotTable, simply click the PivotTable button in the Reports group of the Home tab. Next, click OK on the Create PivotTable dialog. Excel now comes to the foreground with a standard, blank PivotTable1 and a PowerPivot Field List built from PowerPivot. Figure 12–20 displays the initial Excel worksheet.

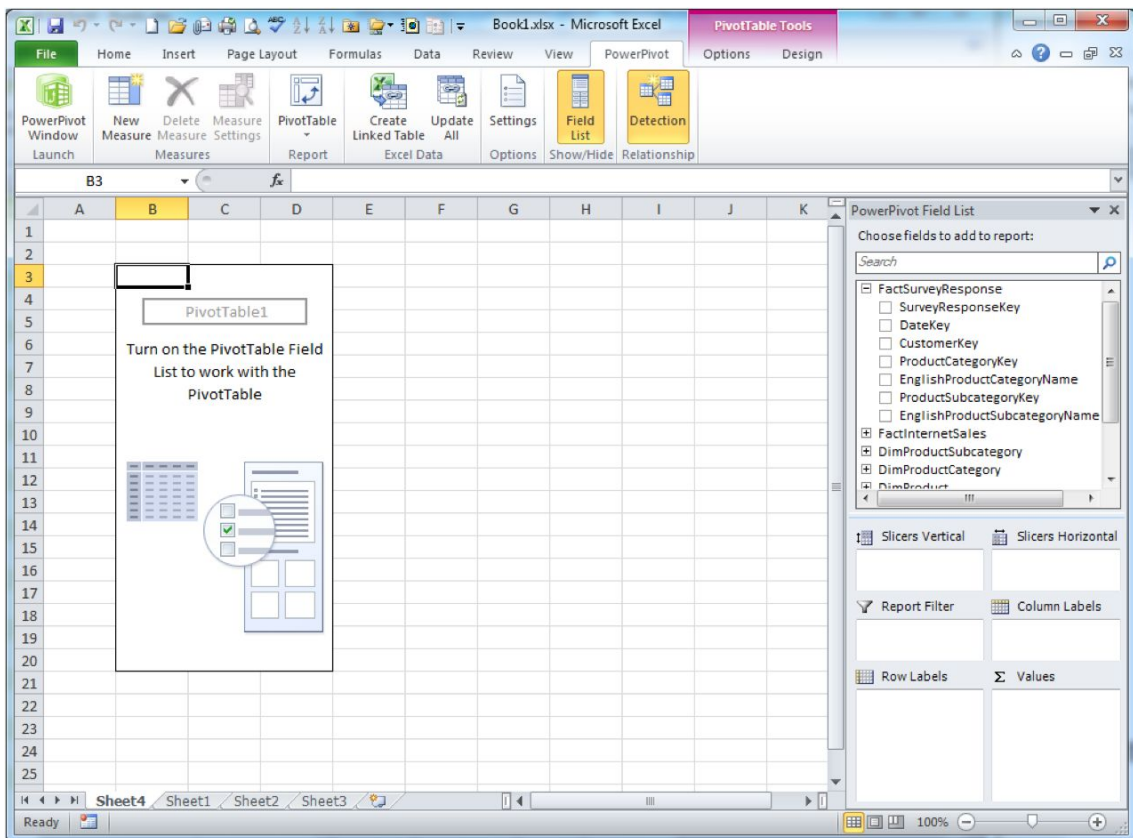


Figure 12–20. An Excel PivotTable and PowerPivot Field List

Next, fill your PivotTable with Adventure Works data. From the PowerPivot Field List, place the EnglishProductCategoryName in Column Labels, and OrderDate (the calculated column you created above) in Row Labels. For the Values, use SalesAmount, and the GrossMargin column you created.

The slicers will come from DimDate. Choose CalendarYear and MonthNumberOfYear from DimDate, and place them in the Slicers Vertical pane. When you choose slicers, Excel places them next to your PivotTable with no filters set.

To use the slicers you just created, select 2008 from the CalendarYear slicer, and 7 from the MonthNumberOfYear slicer. Your completed PivotTable should resemble Figure 12–21.

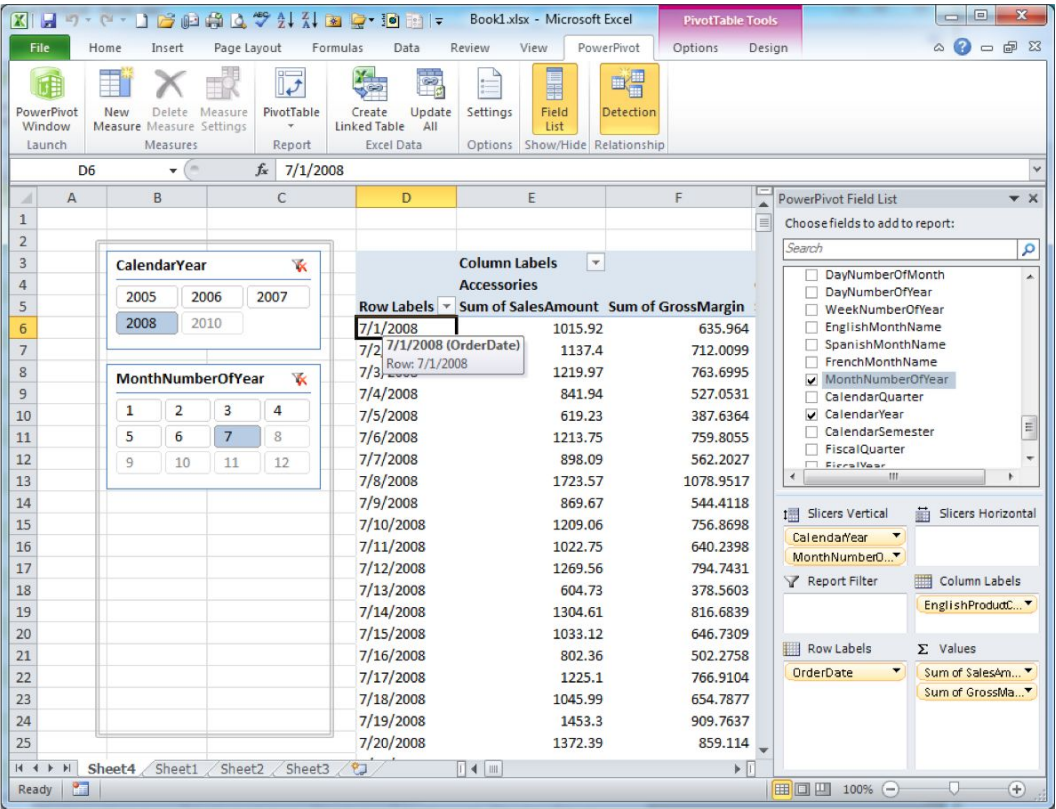


Figure 12–21. An Excel PivotTable using slicers and enriched data from PowerPivot

Summary

This chapter gave you an introduction to the PowerPivot GUI and walked you through using the PowerPivot tool with Adventure Works. In addition, you enriched your PowerPivot data with custom calculations and placed these results into an Excel PivotTable. Finally, you experimented with slicers, which allow data analysts and other end-users to dynamically filter their PivotTable data.

In Chapter 13, you will explore Microsoft’s query language for SQL Server Analysis Services (SSAS) cubes, called Multidimensional Expressions (MDX).



Introduction to MDX

In this chapter, you'll work with Multidimensional Expressions (MDX), which is the native query language for SQL Server Analysis Services (SSAS). MDX is a SQL-like language used to query SSAS cubes.

Although there will be situations that will require that you write MDX expressions or queries from scratch, the design of the BIDS interface really minimizes your need to author much of the MDX that you'll need for your BI projects. A large number of MDX templates, samples, and wizards are available in BIDS and SSMS for you to use and customize.

Unlike working with relational databases where you are probably accustomed to authoring many T-SQL scripts, when working with SSAS cubes, you need more of a reading than a writing knowledge of MDX for typical BI projects. Given that framework, in this chapter we'll cover the following:

- Understanding basic MDX query syntax.
- Adding calculated members, script commands, and named sets to your cubes.
- Understanding the most commonly used MDX functions.

WHY ANOTHER LANGUAGE

MDX is designed specifically to retrieve data from multi-dimensional databases. While a SQL query will return rows and columns, an MDX query returns a cube. To retrieve data, both SQL and MDX use SELECT/FROM/WHERE clauses in their statements. What makes MDX different from SQL is how data is referred to.

A SQL SELECT statement refers to the column names in a table, but an MDX query refers to levels in a hierarchy. Unlike SQL, MDX understands dimensional data, from hierarchy root to leaf level. MDX can return specific levels in a hierarchy as well as related levels, including ancestor, descendant, and sibling levels.

Another MDX advantage is in the area of time reporting. MDX has built-in time-handling functions, including week-to-date, month-to-date, quarter-to-date, and year-to-date. The `ParallelPeriod` function in MDX is suited to financial reporting, allowing year-over-year or other period-over-period comparisons.

MDX Query Syntax

Your journey to understanding MDX syntax begins with understanding the core terminology of Analysis Services. We'll review that terminology first. Next, you'll learn about the basic syntax used in MDX

queries. Once you have a grasp of the syntax, you'll be ready to write your first query. Then you're on the road to discovering business intelligence from your data.

Understanding the Core Terminology

To understand basic MDX syntax, the following are some core OLAP terms used in SSAS:

- *Cube*: Also called an implementation of the Unified Dimensional Model (UDM), this is the core structure in SSAS. MDX also contains the concept of subcubes, which, as the name suggests, are subsets of your enterprise cube. You explored the UDM in Chapter 2.
- *Dimension*: In SSAS, a dimension is a collection of attributes. You can think of it as being based on a group of columns from one or more source tables. The default member of a dimensional attribute is the first member loaded into the dimension. For example, for a customer's dimension like last name attribute, the first value would be alphabetized, so customer "Aand" would appear first by default. You can adjust the default member for a dimensional attribute for all users of the dimension, or you can adjust this value for members of particular SSAS security groups.
- *Hierarchy*: This is an optional grouping of dimensional attributes into a rollup grouping. Hierarchies can either be *regular* (or *pyramid-like*), which means that as you travel downward, each lower level has progressively more members. Hierarchies can also be *ragged*, which means that as you travel downward, lower levels don't always have more members.

An example of a ragged hierarchy can be found in some Geography dimensions. You may have a hierarchy that consists of the following levels: All, Country, State, and City. You may have data in your dimension source table for particular countries that do not have any states but do contain cities.

You can create as many hierarchies of dimensional attributes as are needed to satisfy the business requirements of your project. It is quite common to have several hierarchies in certain dimensions, such as time, where you might have a Calendar Time, Fiscal Time, or a 4-5-4 (retail) Time.

- *Level*: This is a position in the hierarchy. For example, in a time dimension, it is typical to have a Year, Quarter, Month, and Day level. You could, of course, also have a Half-Year level or a Week level in your particular time dimension. You can think of levels as being based on column names from the source table or tables for the dimension. The default level for every dimension is the top (or All) level. You can adjust the level that appears by default—either for everyone or for members of particular security groups.
- *Data member*: This is the actual dimensional data value for an attribute or the member of a level. Another way to think of this is as a particular cell value in your cube. For example, "July 25, 1975" is a member of the time dimension at the Day level. As mentioned earlier, by default, the first member loaded into an attribute value is the one that all end users will see in their dimension and cube browser.

- *Measure*: In the world of MDX, a measure is just another dimension. It does, however, have the following three particular characteristics:
 - The first is that the measures dimension has containers called *measure groups* to associate measures of a similar type or from a particular source fact table together. These measure groups are *not* levels; they are simply display containers.
 - The second characteristic is that the measures dimension is always flat; that is, it never has any hierarchies. A couple of examples of measures would be “Internet Gross Sales” and “Internet Freight Costs.” There is, of course, an All level, but this has little meaning in the measures dimension and is hidden by default.
 - The third characteristic is that nearly all cube browser clients display measures by default in an area called data, rather than on the rows, columns, or filter areas, as other dimension data are displayed. So, measures have a particular default display location in most pivot-table style OLAP client applications. Also, as with other types of dimensional attributes, the first measure created is the one that is displayed in client applications by default. As with other attributes, this can easily be adjusted either globally or for particular groups of end users. You should also remember that measures originate from facts (or rows) in one or more source fact tables. Frequently the terms *measures* and *facts* are used interchangeably.

Figure 13–1 shows a sample dimension to help illustrate some of these terms. The dimension name is DimDate, the hierarchy name is Calendar Date, the selected level name is the Calendar Quarter Description level, and the selected level member is Quarter 3. Just below Quarter 3, you can also see the Month and Day levels, with the Data Member value 2005-08-01 being the lowest level value viewable in the figure. It’s important to understand the terms defined in this section—*dimension*, *hierarchy*, *level*, and *member*—when beginning to write MDX queries.

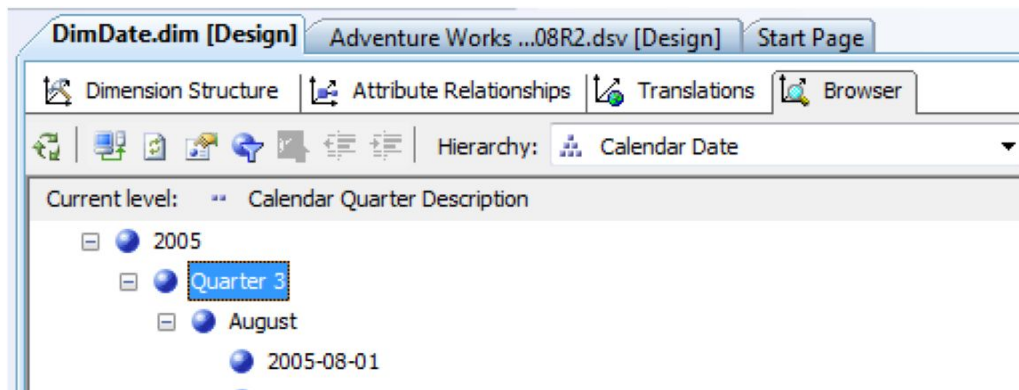


Figure 13–1. The Date dimension, the Calendar Date hierarchy displayed

Learning the Basic Syntax

Your next consideration when working with MDX is to understand its basic syntax. MDX is not case sensitive nor space sensitive. You can add comments to MDX script commands by using the `--` (double dash) to comment out single lines or the `/*...*/` pair to comment out multiple lines, used at the beginning and end of the commented section.

The SQL Server Management Studio (SSMS) has an integrated MDX query tool. To access this tool, you connect to SSAS using SSMS, and then click the Analysis Services MDX Query button on the toolbar. You also have access to a set of MDX query templates in SSMS. The MDX Template Explorer is opened by clicking View on the toolbar and selecting Template Explorer. The MDX Template Explorer is shown in Figure 13–2.

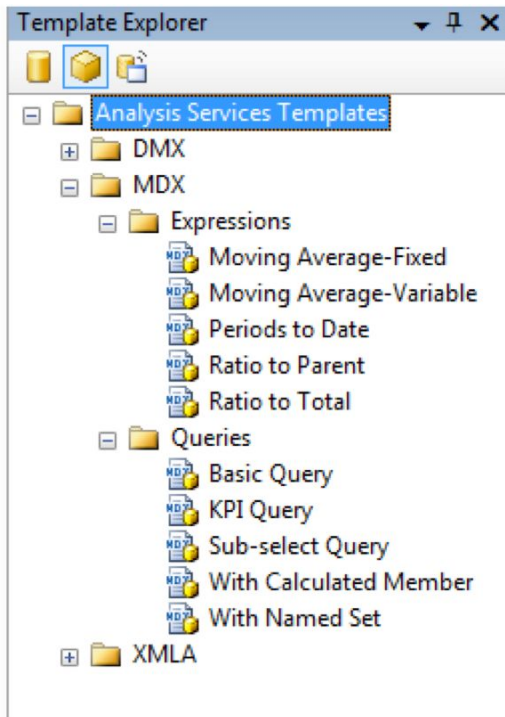


Figure 13–2. The MDX Template Explorer with commonly used MDX sample queries and expressions

Note What is the difference between an MDX query and an MDX expression? An MDX query returns a resultset, which is called a *cellset* in MDX. Cellsets are displayed in a tabular format in SSMS. An MDX expression evaluates to one or more values. Expression templates in SSMS create objects such as *calculated members*, and those calculated members are included in an MDX query as part of the template—so that you can view the result of the expression. Calculated members will be explained in the “Defining Calculated Members” section of this chapter.

Another useful feature of the SSMS environment is the SSAS metadata browser. This allows you to drag and drop cubes, dimensions, hierarchies, levels, or members into MDX queries or query templates. It is nearly always quicker to drag and drop dimension, hierarchy, level, and member names to the query surface than to type them because of the object-naming syntax requirements for MDX objects. It can be quite tedious to type object names.

Manually typed object names can also be a source of code errors. One example of this is the use of square brackets to qualify portions of object names. Strictly speaking, qualifiers (or square brackets) are only required if object names contain embedded spaces or are keywords; however, every GUI tool throughout the BI suite puts square brackets around every object name, which is, in fact, a best coding practice. So, if you drag and drop a measure, dimensional attribute, and so on from any of the metadata browser areas to the MDX query design area, the square brackets will always be added automatically for you.

Another example is the peculiarity of MDX that results in object attributes sometimes being listed by name (or string value) and, at other times, being listed by number (or ordinal position). This is determined based on whether or not member values have been designated as unique at a particular level in a dimensional hierarchy. Because of this, dragging and dropping objects from MDX metadata browsers to query windows is a best practice when you are creating MDX queries or MDX expressions. Figure 13–3 shows the SSAS metadata browser included in SSMS.

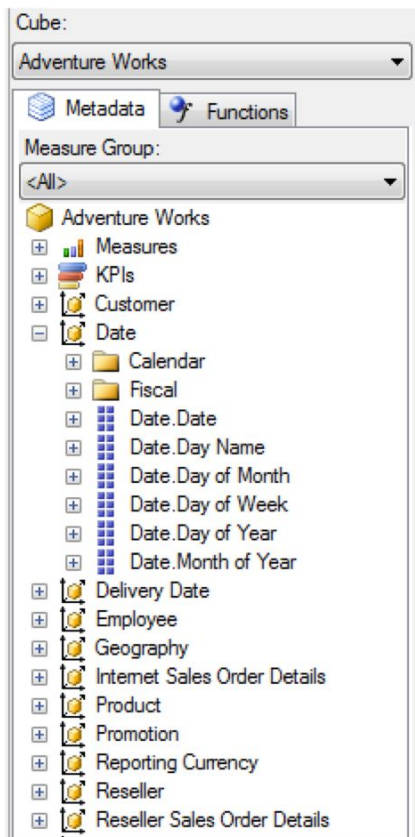


Figure 13–3. The SSAS metadata browser included in the SSMS MDX query window

Now that you've learned a bit about the MDX query environment, you're probably anxious to write your first query. Before you do, however, take a minute to explore the automatic query-writing capabilities of the built-in cube browser (either in BIDS or SSMS). Many common queries are available from the shortcut menu on the cube browser surface. To access the cube browser from within SSMS, right-click any cube in the Cubes folder in Object Explorer and then click Browse.

As mentioned, SSAS requires mostly a reading (rather than writing) knowledge of MDX, so before you jump in to the world of MDX queries, make sure you've exhausted the built-in capabilities of the cube browser in SSMS, which is identical to the cube browser in the BIDS interface. In the cube browser, you can sort, filter, and drill through cube data. A most useful shortcut menu option is Show As. It allows you to see the selected measure in five alternate configurations; Percent of Row Total, Percent of Column Total, Percent of Parent Row Item, Percent of Parent Column Item, and Percent of Grand Total.

Writing Your First MDX Query

When you're ready to write your first MDX query, the best place to start is with the MDX templates. You'll work in SSMS with the MDX interface. Figure 13–4 shows a simple query. You'll start with the FROM portion of the example. FROM most generally takes a cube name as a source. The FROM clause can also reference a subcube; that is written in the form of a nested SELECT statement.

Next is the SELECT clause. Note the on Columns and on Rows clauses—these clauses take axis values and are used to “flatten” the results (or cellset) into a table structure. You can include up to 128 axes in the SELECT clause. If you include multiple hierarchies on the same axis, then the resultset will display those results combined on whichever axis (that is, columns or rows) you've chosen to combine them on. You will sometimes see queries that use the shortcut syntax on 0 (for columns) and on 1 (for rows). You must, however, reference axis values in order; that is, if you want to use an on Rows clause, then you must include an on Columns clause in your query.

The MDX WHERE clause slices the cube along an axis. You can visualize this slice as literally a slice (or portion) of a cube. In the query shown in Figure 13–4, the returned measure value is being restricted to a sum of only those Products that exist in the Product Line name that start with the letter “R.” You can have more than one member in a slice, but each member must come from a different dimension. The WHERE clause member illustrates the member-naming concept discussed in the previous section; that is, the member is listed by ordinal or position in the list of attribute members that starts (at the default ordering) or the letter “R” (&[R]) rather than by actual full name.

You may have expected the rows and columns in this query to show a list of all members in the Customer and Delivery Date dimensions. However, what's actually being shown is the default member of those dimensions (which is, in this case, the All member).

■ **Note** The default member of the measures dimension is being shown. The default member of the measures dimension is the first measure added to the cube. You can adjust the defaultMember property of the measures dimension to something other than the first member added to the cube, or you can include a specific member name in your query, if that is what you want to see returned in your query.

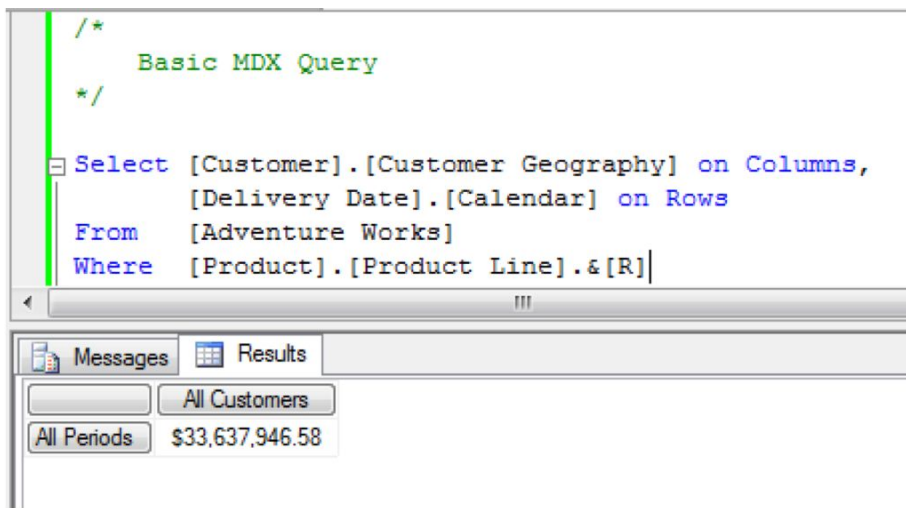


Figure 13–4. A simple MDX query using SSMS

Now that you’ve seen a basic MDX query, let’s create queries to return a list of members, rather than the (single) default member, from a dimension as a result of your query. To do this, you need to understand two additional concepts: tuples and sets.

Discovering Members, Tuples, and Sets

In the “Understanding Basic MDX Query Syntax” section of this chapter, you learned that a data *member* is an item in a dimension of a cube. In MDX, a member is designated by listing the position of the item in the dimension members between square brackets with periods separating the levels in the dimensional hierarchy. Here’s an example:
`[Time].[Years].[2004].`

Tip Square brackets surrounding object names are required only for certain conditions, such as object names that have spaces in them and object names that use reserved keywords. However, it’s common in MDX queries to put square brackets around *all* object names. We suggest that you follow this guideline as a best coding practice.

A *tuple* is a location or a cell in a cube made up of one or more members from each involved hierarchy. In MDX, a tuple is designated by listing the position of all involved dimension members between parentheses, with each member separated by a comma. Here’s an example:

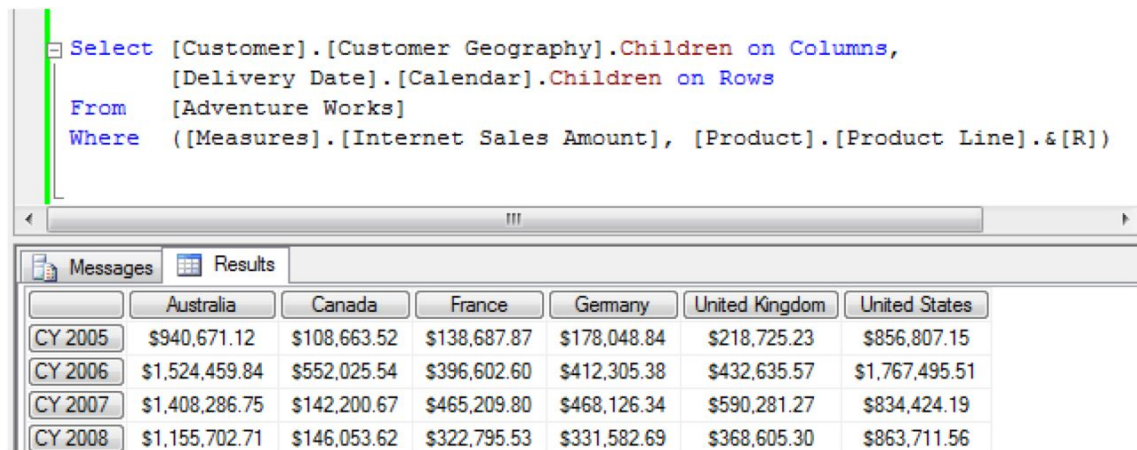
```
([Time].[Years].[2004], [Measures].[Periods To Date])
```

If you do not specify a complete tuple in an MDX query, the default member of the dimension will be returned, which is why the default member of all three dimensions is returned in Figure 13–4.

A *set* is a defined portion or subset of a cube that consists of one or more ordered tuples from different dimensions, surrounded by curly braces and separated by commas. Here is an example:

```
{([Time].[Years].[2004]), ([Product].[Product Line].&[R])}
```

Sets can be created by listing tuples, by using MDX set-generating functions, or by some combination of both techniques. If you create a set by using an MDX function rather than by explicitly listing the members, then you do not need to surround your set with curly braces. In Figure 13–5, you can see a modified version of the original query using the MDX CHILDREN function. Note that the resultset now displays the data members directly below the top level of the hierarchy for both the [Customer].[Customer Geography] and [Delivery Date].[Calendar] dimensions.



```
Select [Customer].[Customer Geography].Children on Columns,
       [Delivery Date].[Calendar].Children on Rows
From   [Adventure Works]
Where  ([Measures].[Internet Sales Amount], [Product].[Product Line].&[R])
```

| | Australia | Canada | France | Germany | United Kingdom | United States |
|---------|----------------|--------------|--------------|--------------|----------------|----------------|
| CY 2005 | \$940,671.12 | \$108,663.52 | \$138,687.87 | \$178,048.84 | \$218,725.23 | \$856,807.15 |
| CY 2006 | \$1,524,459.84 | \$552,025.54 | \$396,602.60 | \$412,305.38 | \$432,635.57 | \$1,767,495.51 |
| CY 2007 | \$1,408,286.75 | \$142,200.67 | \$465,209.80 | \$468,126.34 | \$590,281.27 | \$834,424.19 |
| CY 2008 | \$1,155,702.71 | \$146,053.62 | \$322,795.53 | \$331,582.69 | \$368,605.30 | \$863,711.56 |

Figure 13–5. This query uses the MDX CHILDREN function to display the members of a set.

SSMS includes an MDX function browser in the MDX query tool interface. This helps you to work with the most commonly used MDX functions. Note that the function browser does not contain all MDX functions. If you click and drag any MDX function to the MDX query surface, you'll see the function and any arguments that it takes. Arguments are shown between chevrons (<<and>>). Optional arguments are enclosed in square brackets, such as [<<optional argument>>]. MDX functions are color-coded in brown text on the MDX query surface; MDX keywords are colored blue. The SSMS function browser is shown in Figure 13–6.

Tip You may use IntelliSense in the query window by clicking Edit ► IntelliSense ► List Members. After you select a function, a tooltip will pop up on the function browser with a brief description of what the function does.

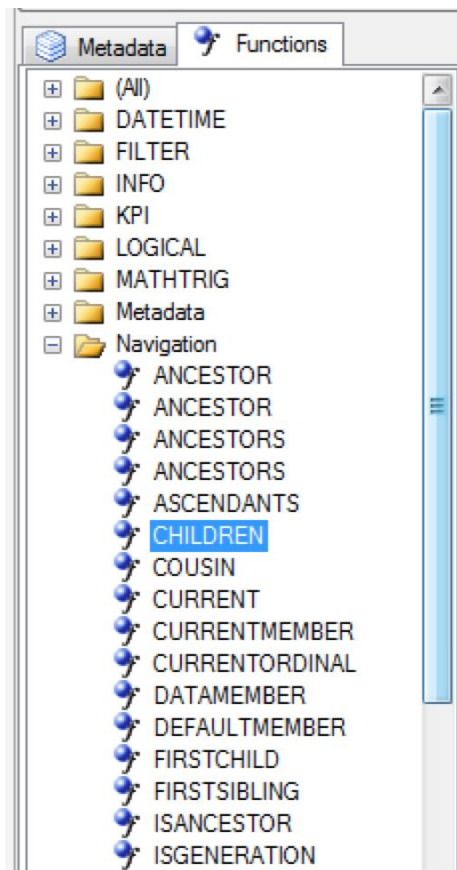


Figure 13–6. The SSMS MDX query area includes an MDX function browser. When you drag the function to the query surface, an argument list will be generated.

Although authoring and executing MDX queries in SSMS is helpful to learning MDX syntax, this is not the most common production task you'll use in your BI solutions. In the next section, you'll learn about the Calculations tab in BIDS, which is where you'll do most of your MDX work. Here you'll add MDX calculated members, named sets, or script commands.

Calculated Members, Named Sets, and Script Commands

In this section, you will work with Calculated Members, Named Sets, and MDX Script Commands. Each of these allows you to work with, and enrich your cube data in different ways. The following list gives a brief definition of each of the features you'll be using in this section:

- **Calculated members:** These are MDX expressions that are evaluated at runtime and can be defined at any level within a hierarchy. Calculated members can also be added to the measures dimension.

- *Named set*: A named set is an MDX expression that returns dimension data and is analogous to a view in relational databases.
- *MDX scripts*: These are batches of MDX that can be used to populate a cube, or part of a cube, with calculations.

Adding MDX Objects to Your Cube

To access the interface for adding MDX objects to your SSAS cube, follow these steps:

1. Open the sample AdventureWorks cube in BIDS by clicking File ► Open Analysis Services Database.
2. In the Connect to Database dialog box, click your servername and database names (AdventureWorks DW Standard [or Enterprise] Edition), and then click OK.
3. Double-click the AdventureWorks cube in Solution Explorer.
4. Click the Calculations tab in the cube designer.

You will work with three panes in this tab: the Script Organizer, the Calculation Tools, and the (MDX) Script Designer. The Script Organizer contains a list of script objects listed in the order they will run in your cube. The Calculation Tools pane contains the useful metadata browser, function references, and templates (similar to what you already worked with in SSMS). The MDX Script Designer pane has two views: a guided GUI interface for calculated members (called Form View) and named sets, and a native script window (called Script view) for any of the three types of MDX objects—calculated members, named sets, and MDX script commands. Figure 13–7 shows the Script Organizer section.

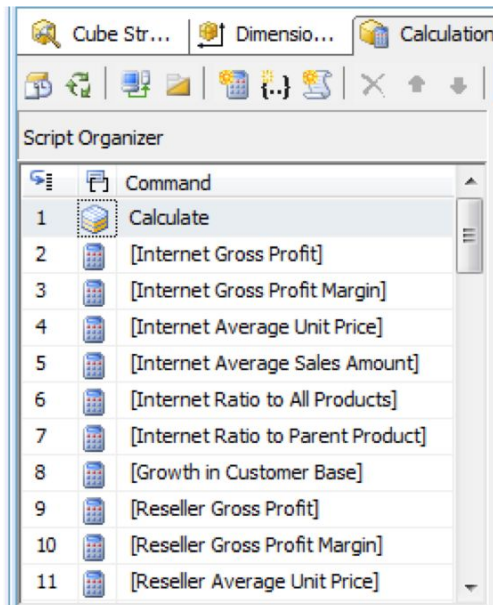


Figure 13–7. The Script Organizer, as seen on the Calculations tab of the cube designer

Calculated members are very similar to calculated cells in Excel workbooks in that an expression (in your case, an MDX expression) returns a value that can be displayed for the end user. The key difference is that the MDX expression returns a set of members rather than a single value.

Similar to cell expressions in Excel, MDX expressions can be created by simply adding, subtracting, and so on existing member values (or tuples). MDX functions also can be used. An example of a calculated member expression (to determine Net Profit) that contains only existing members is

```
[Measures].[Gross Sales] - [Measures].[Expenses]
```

An example of a calculated member expression (to determine an aggregated or combined value for a certain amount of time periods to date) that contains two MDX functions (Aggregate and PeriodToDate) is a bit more complex.

```
Aggregate (PeriodsToDate([<<Target Dimension>>],[<<Target  
Hierarchy>>],[<<Target Level>>],[<<Target Dimension>>].  
[<<Target Hierarchy>>].CurrentMember),[Measures].[<<Target Measure>>])
```

Here's how it works. The Aggregate function combines all results. The PeriodsToDate function takes two dimensions and hierarchies as arguments. For the first argument, you must also supply the level from the dimension that you are interested in, that is, [Time].[CalendarTime].[Quarters]. For the second argument, you must also specify a member. Usually, you are interested in the currently viewed member, so you just use the CurrentMember function. For the last argument, you specify which measure you want to aggregate.

Despite the power of MDX (and a certain subset of unique MDX functions), you'll find that some MDX functions are actually similar to those found in Excel (for example, Sum, Mix, Max, Avg).

The Calculation Tools work area in the Calculations tab in BIDS includes an MDX Function reference area and Metadata areas (identical to the ones found in SSMS when executing an MDX query). It also includes a Template area. The MDX templates in BIDS are different from the ones available in SSMS. In BIDS, the MDX templates are examples of calculated measures or named sets (rather than MDX queries as are available in SSMS), and the BIDS templates are organized by business scenario. The Templates tab of the Financial section of the Calculation Tools area in BIDS is shown in Figure 13-8. All of these templates are examples of calculated measures, as indicated by the small calculator-shaped icon next to each one.

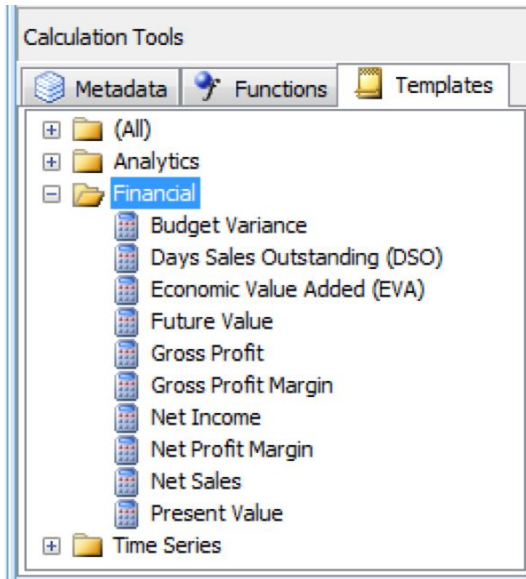


Figure 13–8. The Calculation Tools section on the Calculations tab includes templates for calculated measures.

To take a closer look at the MDX syntax associated with a calculated member, you can double-click the first example in the Script Organizer section of the Calculate tab in BIDS. This will load the MDX associated with the sample calculated measure called Internet Gross Profit into the Form View window. You'll see that calculated measures consist of a couple of items.

- *Parent properties:* This indicates in which dimension the member is to be calculated. Most often, this will be the measures dimension. Because the measures dimension is flat (that is, never has any hierarchies associated with it), you'll leave the parent member property blank in this case.
- *Expression:* This is the MDX expression that will be evaluated to produce the resultset. In this case, Internet Gross Profit, you are using a simple subtract between two other members of the measures dimension.
- *Additional properties:* These include formatting, empty display behavior, and visibility. Also, you can further format by adding information to the color or font expressions areas.

Using Calculated Measures

The most interesting question when working with calculated measures is when you should use them. To answer that, consider the following questions:

- *How frequently will this measure be accessed?* If only a small subset of end users needs access to a measure, consider using a calculated measure. Performance for calculated measures is very good—much faster than SQL Server retrieving calculated column information out of an OLTP store; however, be aware that calculated measures will not be as fast as retrieving information from a stored measure. If the majority of the end users will access a measure that is not present in the source data, it is preferred to calculate the new measure value during the ETL process, store it in the star schema (fact table), and load it into the cube during cube processing.
- *Am I concerned about increasing cube size?* Storing additional information will add to cube storage space time; using calculated members will not add to it.
- *Am I concerned about increasing cube processing times?* Storing additional information will add to cube processing time; using calculated members will not add to it.
- *Do I need measures that do not aggregate (like averages)?* Calculated measures (unlike stored measures) do not aggregate. A common use of calculated measures is to produce average values for a source set.
- *Do I wish to add the complexity of calculating and storing measures to my ETL processes?* An alternative to creating calculated members by writing MDX expressions is to generate the calculated values during the ETL process, store them in the star schema, and load them into the cube.
- *What is my level of proficiency with MDX?* Although Microsoft does provide some calculated member templates, the reality is that if you intend to add a large number of calculated members, you'll probably have to become pretty familiar with the vagaries of MDX. It is important that you accurately assess the time you'll spend authoring the MDX expressions if you choose to use this approach.

We've found calculated members to be quick and easy to add cubes and pretty solid in the performance department. In the BI projects we've been involved with, all have used at least a couple of calculated measures, and some of the projects have used tens or even hundreds of them. Figure 13–9 shows an example of a calculated member from the sample AdventureWorks cube in the BIDS Calculations tab, showing just the calculated member designer.

```

/* Sales Quota Allocation */

/* Allocate equally to quarters in H2 FY 2009 */
Scope
(
    [Date].[Fiscal Year].&[2009],
    [Date].[Fiscal].[Fiscal Quarter].Members,
    [Measures].[Sales Amount Quota]
) ;

This = ParallelPeriod
(
    [Date].[Fiscal].[Fiscal Year], 1,
    [Date].[Fiscal].CurrentMember
) * 1.35 ;

```

Figure 13–9. The calculated member designer

If you want to view or edit the complete MDX statement that the calculated member UI builds for you, click the small Script View button on the toolbar just below the Calculations tab. You will then be presented with the MDX script for not only the calculated measure that you have been working with but also for all items listed in the Script Organizer window. Figure 13–10 shows the script for a calculated measure.

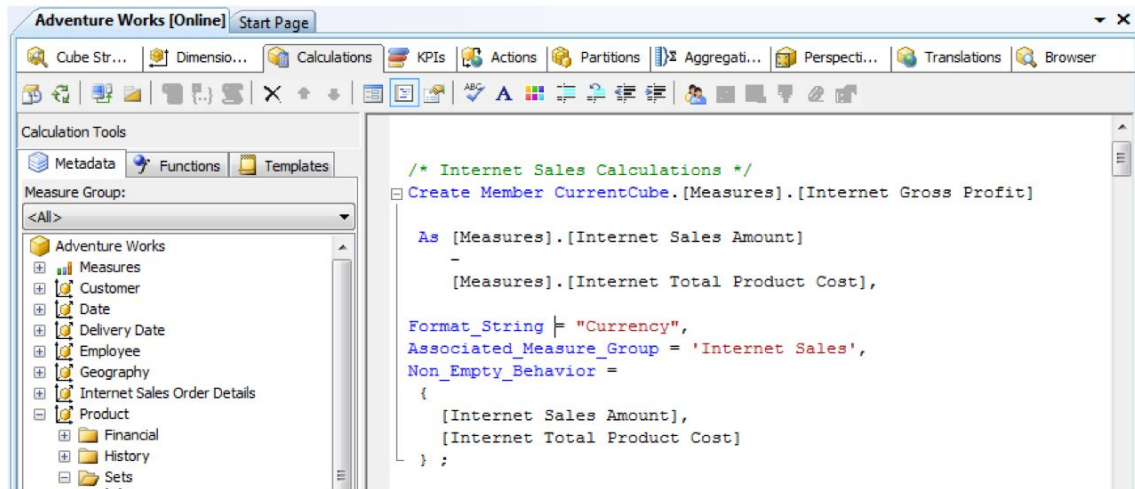


Figure 13–10. If you click the Script View button in BIDS, you can see the MDX script for a calculated measure.

Tip You can use either **With Member** or **Create Member** when you are creating calculated members. The difference is that **With** creates a temporary member, which only lasts for the duration of that particular MDX query, and **Create** creates a durable member—one that can be viewed by any client application that supports calculated members.

Working with Named Sets

Named sets are simply aliases for sets. Sets consist of one or more tuples that are grouped together to define some portion of the cube. Named sets usually contain tuples from within the same dimension. Named sets are most often used to create filters or slices for your queries. In the cube browser, you place them on the (top) filter portion of the interface to use them as filters. A named set called *Long Lead Products* is shown in Figure 13–11.

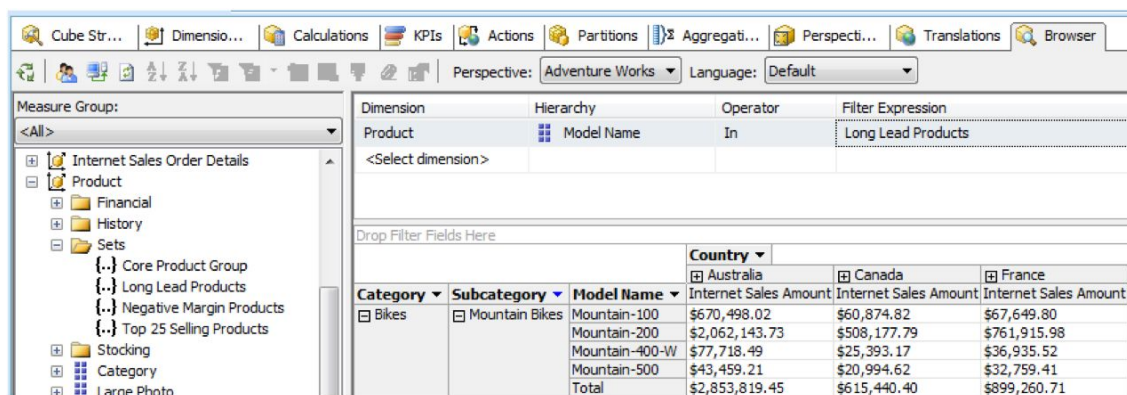


Figure 13–11. Named sets appear in a *Sets* folder under the dimension where they are created. To use them in the cube browser, drag the named set to the (top) filter portion of the UI.

If you want to use your named set as a slicer (WHERE clause), you'll have to write an MDX query. Slicers restrict results to the set values included. We've included a query in SSMS using one of the named sets defined for the AdventureWorks sample cube. You'll note that the named set appears in the WHERE clause of the MDX query statement as shown in Figure 13–12.

```
Select  [Product].[Product Line].Members on Columns,
        [Product].[Days To Manufacture].[4] On Rows
From    [Adventure Works]
Where   ([Measures].[Internet Sales Amount], [Long Lead Products])
```

Figure 13–12. Named sets are created to be used in the WHERE (slicer) or FILTER (which returns the set that results from filtering a set) portion of an MDX query statement.

■ **Note** The MDX `Members` function returns a list of members at the level immediately below the level specified in the query object; that is, for `Product Lines`, the members `mountain`, `road`, and `touring` are returned on the columns axis in the sample.

Named sets are really just a convenience (like relational T-SQL views). They allow you to read a specific subset of a cube. Using them can speed up development because they can be used multiple times to refer to complex definitions. They are unlike relational views in that they are *not* security mechanisms. To set security for specific portions of a dimension, you can use SSAS security roles.

Writing Script Commands

The third type of object that you can add to your SSAS cube via the Calculations tab in BIDS is the MDX script command. This name isn't really accurate, as this feature hasn't been enabled to allow you to add all types of MDX script commands to your cube, rather (usually) one particular type: a script using the MDX `Scope` keyword.

Similar conceptually to a named set, a script with a `Scope` keyword allows you to define a subset of your cube (sometimes called a *subcube*). Unlike a named set, this subcube is usually created so that you can read it and also make changes (or write) to it. To enable writing by end users to any portion of the cube, you must enable both the `Writeback` property and `Writeback` permissions for the particular portion of the cube of interest.

■ **Note** You can also use the MDX keywords `Calculate` (which creates a subset of a cube, or a subcube) or `Freeze` (which locks cell values in a subcube) in an MDX script. For more information, see the BOL topics “The Basic MDX Script” and “MDX Scripting Statements.”

There are two parts to a script command. The `Scope` statement creates the subcube. This function applies whatever change you want to make to the subcube. The sample script called `Sales Quota Allocation` is a good example of using script commands. Switch to the Script view on the Calculations tab in BIDS and you'll see two complete script commands (using both the `Scope` and the `This` keywords) as shown in Figure 13–13.

The most common scenario for subcubes is the one shown in this example: budget allocations based on past history and other factors. In this script, the `Scope` statement defines subcubes, and the `This` keyword applies new values to the named members of the subcube. This example is typical of the reason you would use MDX script commands—to facilitate budgeting “what if” scenarios.

Subcubes are convenient for these kinds of scenarios because business budgeting is typically based on a number of factors, some past known values combined with some future predicted (or unknown) values. These factors often need to be applied to some named subcube of your enterprise data.

```

/* Sales Quota Allocation */
/* Allocate equally to quarters in H2 FY 2009 */
Scope
(
    [Date].[Fiscal Year].&[2009],
    [Date].[Fiscal].[Fiscal Quarter].Members,
    [Measures].[Sales Amount Quota]
) ;

    This = ParallelPeriod
        (
            [Date].[Fiscal].[Fiscal Year], 1,
            [Date].[Fiscal].CurrentMember
        ) * 1.35 ;
End Scope ;

/* Allocate equally to months in FY 2006 */
Scope
(
    [Date].[Fiscal Year].&[2006],
    [Date].[Fiscal].[Month].Members
) ;

    This = [Date].[Fiscal].CurrentMember.Parent / 3 ;
End Scope ;

```

Figure 13–13. The new *Scope* and *This* MDX keywords allow you to create subcubes and write changes to the MOLAP data.

Another consideration when using the Calculations tab in BIDS to design MDX script objects is the order you add the script objects. Script commands are evaluated and then executed in the order (top-to-bottom) listed in the Script Organizer window. You can change the order of execution by right-clicking any one script and then clicking Move Up or Move Down. You can also change the order of execution for calculated members (or cells) by using the MDX keyword `SOLVE_ORDER` inside of the affected script commands.

The final consideration when using the Calculations tab in BIDS to design any of the supported MDX objects (calculated members, named sets, or MDX script commands) is that you now can debug any portion of the resultant script. All MDX objects that you add to your OLAP cube are added by the execution of one, ordered MDX script.

To enable debugging in this script, you must be working with the objects in the Calculations tab in Script view. Once there, you simply click once in the gray margin next to the line of the MDX object for

which you want to set a breakpoint. An example is shown in Figure 13–14. While you are in break mode, you can examine the value of variables, as you do in traditional debugging. Also, BIDS presents you with a series of SSAS-specific “watch” windows, where you can execute MDX queries by dragging and dropping items from the metadata browsers to those debugging windows.

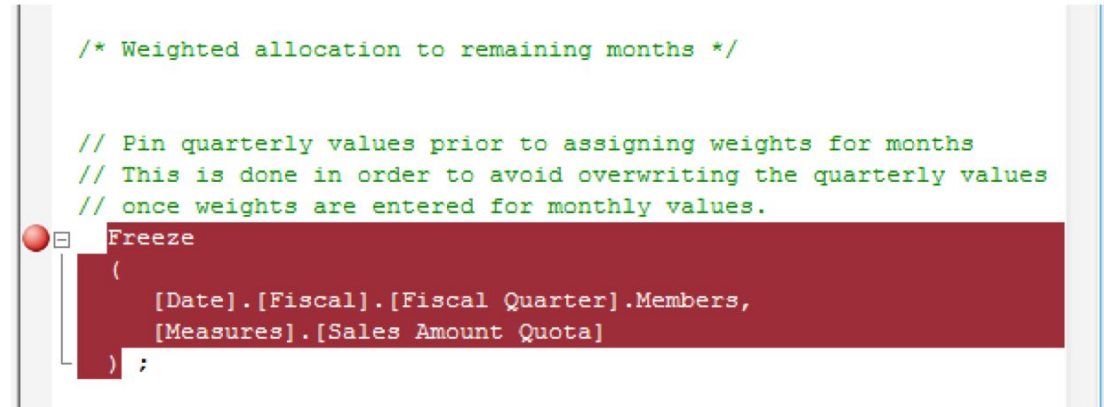


Figure 13–14. Using the Script view in the BIDS Calculations tab, you can set breakpoints in the MDX script associated with your cube.

Common MDX Functions

To give you a flavor for working with MDX, here is a brief list of commonly used MDX functions types, along with any arguments for each function:

Current: This function returns the current tuple from a set. You should note that this function only works with named sets. You can think of this function as a kind of a `CurrentMember` function, but one that is specific to named sets.

DistinctCount: This function returns a count of unique, nonempty tuples in a set.

IIF: This function works similarly to IIFs (Immediate If) in other applications (Excel, for example). It takes three arguments: an expression, a value to be returned if the expression evaluates to true, and an alternate value to be returned if the expression evaluates to false. The IIF function is commonly used in the Status expression for many KPIs, such as the Customer Profitability template (along with many others). You’ll note that KPIs return either 1 (for a positive result) or -1 (for a negative result). The Status expression section of the template, illustrating the IIF function, is shown in Figure 13–15.

The screenshot shows a configuration window for a KPI. The 'Name' field is 'Customer Profitability'. The 'Associated measure group' is 'Internet Sales'. The 'Value Expression' is `[Measures].[Internet Sales-Total Product Cost] - [Measures].[Internet Sales-Product Standard Cost]`. The 'Goal Expression' is `[Measures].[Internet Sales-Total Product Cost] - [Measures].[Internet Sales-Unit Price]`. The 'Status' section has a 'Status indicator' of 'Road signs' and a 'Status expression' that uses the `IIF` function to compare the KPI value to the goal.

Name:

Associated measure group:

Value Expression

Goal Expression

Status
 Status indicator:
 Status expression:

)"/>

Figure 13–15. The MDX function `IIF` is commonly used when creating KPIs.

Filter: This function returns a subset of the listed set based on the action of a filter (which contains a condition). Using this function makes your queries more efficient, as it reduces the amount of data being returned. The Filter function is used in MDX queries to restrict data on one of the axes, as shown in the following query:

```
With
  Set [Big Sales] As
    {[Measures].[Internet Sales Amount],
     [Measures].[Reseller Sales Amount]}

Select
  [Big Sales] On Columns,
  Filter(([Delivery Date].[Fiscal].Members),
    [Geography].[Geographies] <> 'Australia') on Rows
From
  [Adventure Works]
Where
  [Product].[Product Line].&[R]
```

Generate: This function has two types of functionality, which are shown in the argument lists in Figure 13–15. The first functionality applies the second set to each member of the first set in the argument list and then joins the resulting sets by union. The Generate function eliminates duplicates by default but does provide you with the option of including them (by specifying the optional “all” third argument).

The second functionality returns a concatenated string created by evaluating a string expression over a set. You have the option, using the third argument, to specify a particular delimiter in the results.

The *Generate* function is often used when creating calculated members to write MDX queries that perform operations on multiple sets in a more concise way.

SetToArray: This function converts a set (or sets) to an array. The optional arguments are the second set and the numeric expression. The return value, which is the datatype variant (VT_ARRAY), should only be used as input to a user-defined function.

ParallelPeriod: This function returns a member from a prior time period (year, quarter, month, etc.) in the same relative position as a member listed. It is often used with the MDX *CurrentMember* and *Parent* functions. An example of this function is shown in one of the scope script commands provided with the AdventureWorks sample. This script is shown in the following query:

```
// Scope on month level in FY 2007 and onwards
Scope
(
    [Date].[Fiscal Year].&[2007] : Null,
    [Date].[Fiscal].[Month].Members
) ;

// Compute weights based on reseller sales ratio in previous year
This =
(
    ParallelPeriod //Fetch reseller sales amount in previous year
    (
        [Date].[Fiscal].[Fiscal Year], 1,
        [Date].[Fiscal].CurrentMember
    ),
    [Measures].[Reseller Sales Amount]
)
/
(
    ParallelPeriod //Divide monthly value by quarterly value to obtain ratio
    (
        [Date].[Fiscal].[Fiscal Year], 1,
        [Date].[Fiscal].CurrentMember.Parent
    ),
    [Measures].[Reseller Sales Amount]
) ;

// Allocate quarterly values to months according to weight
This = [Measures].CurrentMember * [Date].[Fiscal].Parent ;

End Scope ;
```

Aggregate: This function has been enhanced to work with *DistinctCount* measures, as well as with the various semiadditive measures. Note that numeric expression is an optional argument.

Freeze: This statement (not function) locks the specified value of the current subcube to the specified values. It is used in MDX script commands to “pin” a subcube (or exempt it from being updated) during the execution of a script using the *Scope* statement and *This* function. In the example shown in Figure 13–20, the *Freeze* statement is used to lock the current quarterly values of

the Sales Amount Quotas in a cube that uses a Scope/This statement to assign new values (weights) to the Sales Amount Quotas at the month level. Without the Freeze statement, these updates would aggregate up to the Quarter level, which, in this particular business scenario, would be undesirable. The following query uses the Freeze statement:

```
Freeze
(
    [Date].[Fiscal].[Fiscal Quarter].Members,
    [Measures].[Sales Amount Quota]
);
```

Exists: This function takes one or more sets that can be filtered and returns a set of members that exist within the other sets. The [Filtered Set of Attribute Members] named set template illustrates the use of the Exists function.

```
Exists
(
    [Promotion].[Promotion].[Promotion].Members,
    Filter
    (
        [Promotion].[Discount Percent].[Discount Percent].Members,
        [Promotion].[Discount Percent].CurrentMember.MemberValue >= .30
    )
)
```

KPICurrentTimeMember, KPIGoal, KPIStatus, KPITrend, KPIValue, KPIWeight: These functions are used in the KPI tab in BIDS when adding KPIs to your cube. They all basically function as aliases for MDX (usually CASE) expressions that are defined in the KPI builder area of BIDS.

MemberValue: This function allows you to directly retrieve (or use for comparison) a value associated with a dimension attribute (configured via the ValueColumn property). The ValueColumn page for a dimensional attribute is found on the dimensional attribute's properties pane.

This: This function allows you to apply an update to the MOLAP data as defined in a subcube (usually by using the MDX keyword Scope to define the subcube).

UnknownMember: This returns the unknown member value for a level or member. Remember that you have the ability to configure an unknown member value for an entire dimension by setting the property values for both the UnknownMember and the UnknownMemberName.

Unorder: This function allows you to improve the efficiency of MDX queries by specifying that you have no need for specifically ordered results. Some MDX functions use Unorder automatically (Sum, for example). If you would like to verify whether a function you are using does this, you can open a trace in Profiler to see the internally generated MDX query.

Summary

In this chapter, we reviewed basic MDX syntax and querying. We then discussed the core objects that you might add to an SSAS cube: calculated members, named sets, and script commands. We next reviewed the most commonly used MDX functions and some of the new ones as well. The last area we covered was the new and powerful capability to associate assemblies with SSAS.

In the final chapter, we'll explore the toolset available in SSAS for data mining.



Introduction to Data Mining

In this chapter, we'll explore the incredibly powerful tools included with SQL Server Analysis Services (SSAS) for use in data-mining solutions. You can begin by thinking of data mining as a terrific "value add" to your BI solution.

Data mining in SSAS includes a number of easy-to-use wizards. These wizards contain consistently and remarkably well-documented dialog boxes. However, data mining *is* a complex topic, and it's important to understand that this chapter is an introduction to this topic.

■ **Tip** Microsoft's data-mining team has a web site devoted to data mining with SQL Server, which includes tips, notes, and samples: www.sqlserverdatamining.com.

The SSAS data-mining facilities are among the brightest of the gems in the entire BI feature set because of the included power and the improved ease of implementation. We've got a great deal of ground to cover here, and this is what we'll discuss in the chapter:

- Defining SSAS data mining
- Reviewing data-mining structures
- Explaining the nine SSAS data-mining algorithms
- Processing mining models
- Using the data-mining language: Data Mining Extensions (DMX)

Defining SSAS Data Mining

Data mining is a set of sophisticated tools and algorithms that allow analysts and end users to solve problems that would otherwise take huge amounts of manual effort or else would simply remain unsolved. You can also think of data mining as a set of tools to help you enable your end users to discover patterns and trends based on defined subsets of enterprise data. Such analysis is useful in many types of business situations, such as predicting future values and better understanding why your business got the past results that it did.

Using SSAS, you can create data-mining structures, which contain mining models. *Mining models* represent implementations of specific data-mining algorithms. *Algorithms* are mathematical functions that perform specific types of analysis on the associated data set. One example is the Microsoft Time

Series algorithm. This algorithm predicts a specific value—for example, rate of sale (or quantity) of a particular item—over time.

These mining models can be built using online analytical processing (OLAP) cubes or on relational tables as source data. Data mining is usually (but not always) meant to be complementary to an SSAS cube. A cube is often used to verify results: that is, “We think this happened; does the data support our belief?” You can use mining structures (and their contained models) to discover correlations, patterns, and other surprises in the data: that is, “what will happen?” Another common use of mining is when businesses buy data from data-collection companies that represents competitive information (this type of data is often called *competitive data*). Data mining can be used to help businesses consider what would happen if they got into a certain type of new business, what would happen if they started doing business in certain locations, and so on.

Some of the focus areas of SSAS data mining are as follows:

- Make data mining easier to implement. Data mining has traditionally been one of the most challenging types of data-analysis solutions to put into operation, due to the need to deeply understand the various algorithms involved. The tools provided in BIDS make the creation of mining structures much easier by including intelligently designed wizards with (mostly) self-documenting dialog boxes. Also, importantly, there are now tools to help you verify the accuracy of your mining model. This can help you select the most useful algorithms.
- Make it easy for end users to work with the results of mining structures in ways that are meaningful to them. To this end, SSAS includes a broad variety of data-mining-model viewers in BIDS, many new types of client integration (discussed in detail later in the chapter), and an API for custom viewer development.

SSAS data-mining methods are expressed as algorithms; nine algorithms are included in SSAS 2008 R2. In addition, the UI and object model for data mining are both powerful and easy to use.

One of the most challenging parts of data mining is understanding what the various algorithms actually do and then creating a mining structure that provides the needed data points to the appropriate algorithm or algorithms that best support your specific business requirements. Also, some algorithms function more effectively after you configure some of their available properties. In addition, understanding how to best configure the algorithm properties can be challenging. We will review some of the more important algorithm properties in this chapter. Another important consideration is how you present the completed data-mining model results to your end users.

To help you select the most appropriate data-mining algorithm or algorithms, we’ll start by thinking about some of the business problems that SSAS data mining can impact. Later, we’ll tie these types of problems to particular algorithms. Here’s a partial list:

- What characteristics do your customers share? How could you group them or put the types of customers into buckets? This type of information could be used, for example, to improve effectiveness of marketing campaigns by targeting different campaign types more appropriately: you could use magazine ads for customers who read magazines, TV ads for customers who watch TV, and so on.
- What situations are abnormal for various groups? This type of analysis is sometimes used for fraud detection. For example, purchasing behavior outside of normal locations, stores, or total amounts might be indicative of fraud for particular customer groups.

- What product or services should be marketed or displayed next to what other products or services? This is sometimes called *market-basket* analysis and can be used in scenarios such as brick-and-mortar store shelf product placement when you're considering which products should be next to each other, or for web marketing when you're considering which ads should be placed on which product pages.
- What will a certain value be (for example, rate of sales per week) for an item or set of items at some point in the future, based on some values (for example, the price of the item) that the item had in the past? An example of this would be a retailer that adjusts the price of a key item upward or downward based on the sell-through rate for that price point for that type of item for particular groups of stores in an effort to control the amount of inventory of that particular item over time in each store.

The best way to start working with SSAS data mining is to explore the models that ship with the sample AdventureWorks solution. To do this, open the AdventureWorks sample in BIDS. Note that the sample includes five data-mining models. We'll use these for the basis of the data-mining discussion in this chapter.

Figure 14–1 shows the sample data-mining containers, which are called *mining structures*, in the Solution Explorer in BIDS. Each mining structure showcases using one or more mining models (with each mining model containing a particular set of algorithms applied to source data) to impact a different type of business problem.

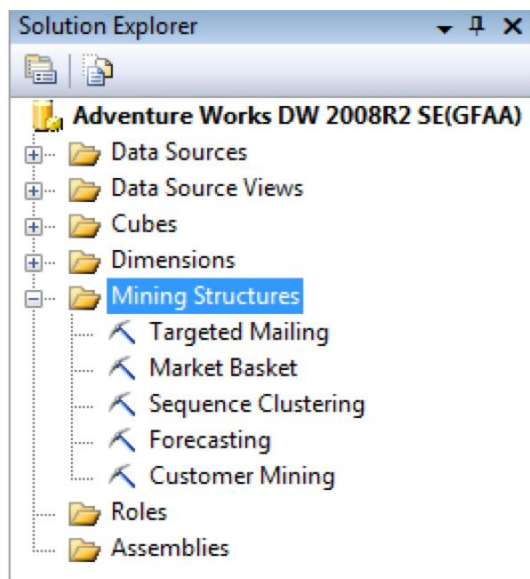


Figure 14–1. The AdventureWorks solution contains five sample mining structures.

Data-Mining Concepts

The Data Mining Wizard simplifies creating a mining structure in SSAS. To start the wizard, you right-click the Mining Structures folder in Solution Explorer in BIDS, and then click New Mining Structure. Your first choice when using the wizard is the location of the source data for the mining structure. Here, you select either a subset of a defined data source view (DSV; which is most often based on relational data) or a dimension from one of the cubes in the same SSAS project. Your next choice is to select one of the nine available data-mining algorithms.

Before we start exploring the individual algorithms, we'll first discuss general categories of data-mining algorithms, the types of business problems those categories are designed to impact, and which SSAS data-mining algorithms are available in which category or categories. A nice feature included in SSAS is the ability to easily add additional mining models (using different algorithms and/or different input columns) to the same mining structure. You may find that using more than one mining model, and reviewing the results of each, is an effective strategy when implementing data mining as part of your BI solution.

Here's a list of categories or types of data-mining tasks:

- *Classification*: This involves predicting the value of one or more fixed variables, based on multiple input variables (or attributes). These types of algorithms are often used when a business has a large volume of high-quality historical data. The included algorithm most often used to implement this technique is Microsoft Decision Trees. The Microsoft Naïve Bayes and Microsoft Neural Network algorithms can also be used. The Naïve Bayes algorithm is so named because it assumes all input columns are completely independent. The Microsoft Neural Network algorithm is often used with very large volumes of data that have very complex relationships. With this type of source data, Microsoft Neural Network can sometimes produce the most meaningful results of all supplied algorithms.
- *Segmentation or Clustering*: This involves grouping source data into categories (sometimes also called *segments* or *buckets*) based on a set of supplied values (or attributes). All attributes are given equal weight when determining the buckets. These types of algorithms are often used as a starting point to better understand the relationships between attributes in a large volume of data. Businesses also use them to make more intelligent “like-for-like” predictions, such as this store is like that store in these categories, and so it should perform similarly in this category. The included algorithm most often used to implement this technique is the Microsoft Clustering algorithm.
- *Association*: This involves finding correlations between variables in a set of data. This is often called *market-basket* analysis. The goal of the algorithm is to find sets of items that show correlations (usually based on rates of sale). It is used to help businesses improve results related to cross-selling. In brick-and-mortar locations, the results can be used to determine shelf placement of products. For virtual businesses, the results can be used to improve click-through rates for advertising. The included algorithm most often used to implement this technique is the Microsoft Association algorithm.

- *Regression or Forecasting:* This involves a process that is similar to classification: predicting a value based on multiple input variables. The difference is that the predictable value is a continuous number. In the case of a forecasting, the input usually contains a time series. Businesses use regression algorithms to rate a sale of an item based on retail price, position in store, and so on or to predict the amount of rainfall based on humidity, air pressure, and temperature. The included algorithm most often used to implement this technique is the Microsoft Time Series algorithm. The Microsoft Linear and Microsoft Logistical Regression algorithms can also be used.
- *Sequence Analysis and Prediction:* This involves finding patterns in a particular subset of data. Businesses can use this type of algorithm to analyze the click-path of end users through a commercial web site. These paths or sequences are often analyzed over time: what items did the customer buy on the first visit? What did the customer buy on the second visit? Sequence and association algorithms both work with instances (called *cases* in the language of data mining) that contain a set of items or states. The difference is that only sequence algorithms analyze the state transitions (the order or time series in which cases occurred); association algorithms consider all cases to be equal. The included algorithm most often used to implement this technique is Microsoft Sequence Clustering.
- *Deviation Analysis:* This involves finding exceptional cases in the data. In the language of data mining, these are often called *outlier cases*. Businesses use this type of algorithm to detect potential fraud; one example is credit card companies that use this technique to initiate alerts (which usually result in a phone call to the end user, asking for verification of a purchase that's particularly unusual due to location, amount, and so on). The most commonly used algorithms for this type of analysis are Microsoft Decision Trees used in combination with one or more other algorithms (often Microsoft Clustering).

Architectural Considerations

To implement data mining in a BI solution, you need to consider your requirements in light of several architectural concerns:

- Determine what type of business problems you want to impact.
- Review the quality of the source data. Is additional ETL warranted to clean or validate that data? These processes may include aggregating source data, removing nulls, and removing abnormal data points (outliers).
- What data should be included in your model? Here you select tables and columns from relational data or dimensions, attributes, and measures from your cube. Will any source data be from related (or nested) tables? What columns will be marked as keys, as inputs, as predictable (more on what these terms mean later in this chapter)?
- Which algorithms will you begin with in your data-mining structure? Remember, it's relatively easy to add algorithms as you continue to work with your project.
- How will you validate the results of your models? Which algorithms prove to be most useful for presenting you with useful information regarding your particular business scenario? Also, which algorithms prove to be most accurate? We'll discuss more about the tools included in BIDS to do this later in this chapter.

- Who will view the results of your mining structure? What client tools will end users use? Will custom application development be required?
- How will your mining model be maintained? How often will new data be added?
- At what intervals will the model be reverified? What metrics or baselines will be used to “prove” the validity and usefulness of your models?

■ **Tip** For more information about lifecycle processes for data mining, you can review a standard model for the data-mining lifecycle. It’s called the Cross Industry Standard Process for Data Mining (Crisp-DM) method and is described at www.crisp-dm.org.

The next step in our journey is to dig a bit deeper into the sample mining structures supplied with the AdventureWorks cube.

Reviewing Data Mining Structures

To begin reviewing your first data-mining structure, right-click the Targeted Mailing mining model, and select Open. The Targeted Mailing model opens to the Mining Structure tab and shows the vTargetMail view in the data source view (DSV) pane and the mining model in the left-most pane.

Mining Structure Tab

The first tab in the work area is the Mining Structure tab. Here you can see the source data included in your mining model (in a DSV-like format). An important difference between this view and the DSV container in a BI project is that you can only view the source data in the mining structure view. You cannot add calculated columns, and you have to use the *original* DSV defined at the level on the SSAS database (as you do when creating a cube) to make any structural changes to the DSV used as a source for your mining structure. If you need to add or remove columns or nested tables, you use the original level view.

You can configure several properties for both the entire source or for individual columns in this view. An example is the CacheMode property. Your choices are KeepTrainingCases and ClearAfterProcessing.

■ **Note** What is *training data*? In this chapter, we’ll often use this term. It means to provide the data-mining model with sample data so that the model can “learn” from the sample cases. In SSAS, *process* is equivalent to *train*.

The latter option is often used during the early development phase of your mining project. You may process the model, only to find that the data used needs further cleaning. In this case, you perform the subsequent clean and then reprocess the model. Figure 14–2 shows the properties for the targeted mailing sample.

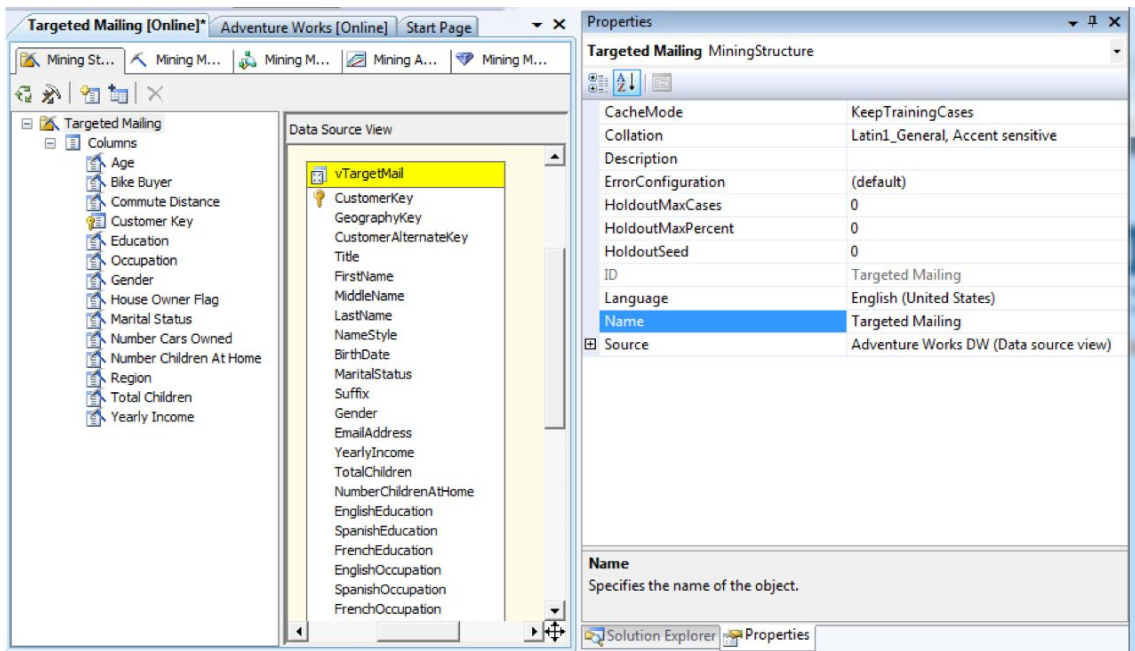


Figure 14–2. In the Mining Structure tab of BIDS, you can view the source data and add or remove columns and nested tables.

Mining Models Tab

The next tab in the mining structure designer in BIDS is the Mining Models tab. Here you view the one or more mining models that are included in the particular mining structure that you are editing. You can easily add new models to your structure by right-clicking the surface and then clicking New Mining Model. Also, you can change the source data associated with a type of mining model by creating more than one instance of that model and ignoring one or more columns from the mining structure DSV. You may choose to do this to test the level of impact that a particular attribute has on a mining model. The Ignore option is shown for the Yearly Income column using the Microsoft Naïve Bayes algorithm in Figure 14–3.

| Structure | TM Decision Tree | TM Naive Bayes |
|-------------------------|---------------------------|-----------------------|
| | Microsoft_Ddecision_Trees | Microsoft_Naive_Bayes |
| Age | Input | Input |
| Bike Buyer | Predict | Predict |
| Commute Distance | Input | Input |
| Customer Key | Key | Key |
| Education | Input | Input |
| Occupation | Input | Input |
| Gender | Input | Input |
| House Owner Flag | Input | Input |
| Marital Status | Input | Input |
| Number Cars Owned | Input | Input |
| Number Children At Home | Input | Input |
| Region | Input | Input |
| Total Children | Input | Input |
| Yearly Income | Input | Ignore |

Figure 14–3. Two mining models, one based on Microsoft Decision Trees and one based on Microsoft Naïve Bayes, are included in the mining structure shown.

You can change the use of the associated (nonkey) source columns in the following ways:

- *Ignore*: This setting causes the model to remove the column from the model.
- *Input*: This setting causes the model to use that column as source data for the model.
- *Predict*: This setting causes the model to use that column as both input and output.
- *PredictOnly*: This setting causes the model to use that column as output only.

Note There are specific requirements about Input and Predict columns for each type of algorithm. These are discussed later in this chapter.

You can configure the algorithm parameters for each mining model in the mining structure by right-clicking the mining model on the design surface and then clicking Set Algorithm Parameters. These

parameters vary depending on which mining algorithm you are working with. Figure 14–4 shows the configurable parameters for the Microsoft Decision Trees algorithm. Note that when you select one of the properties, the configuration dialog box shows you a brief definition of the configurable property value. Figure 14–4 also shows the definition for the COMPLEXITY_PENALTY property.

You may be surprised to see the configuration of parameters being included at this point in this chapter, assuming that such configuration is for advanced users of data mining only. Actually, you will find yourself exploring (and changing) many of the parameters as you begin to tinker with data mining. The UI does a pretty good job of explaining each of them; there is also additional documentation in BOL.

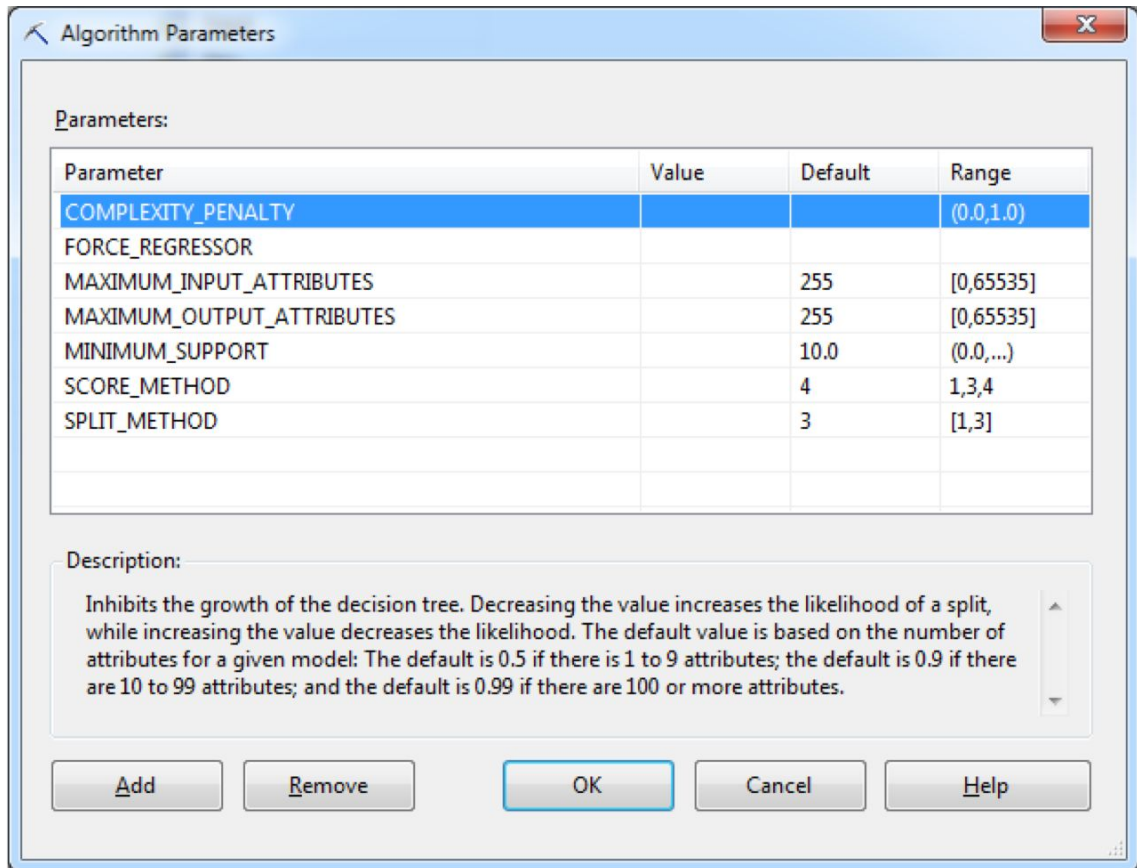


Figure 14–4. BIDS allows you to do additional configuration that is specific to the algorithm type you are working with.

As you become a more advanced user of data mining, for certain algorithms, you may choose to add your own custom parameters (and configure their values) via this dialog box.

Mining Model Viewer Tab

The next tab in the BIDS mining structure designer is Mining Model Viewer. An interesting aspect of this tab is that each mining model algorithm includes one or more types of mining-model viewers. These viewers are for you, as the mining-model designer, and not for end users. The purpose of the broad variety of viewers is to help you to determine which mining-model algorithms are most useful for your particular business scenario.

In this case, look at the included TM Decision Trees sample in AdventureWorks. This model shows information correlated to bicycle purchasing: “number of cars owned,” “number of children at home,” and so on.

The viewers include both graphical and text (rows and columns of data) representations of data. Some of the viewers include multiple types of graphical views of the output of the mining-model data. Additionally, some of the viewers include a mining legend in the properties area of the design surface.

■ **Tip** The mining structure viewers are available in SQL Server Management Studio (SSMS). To access them, connect to SSAS in SSMS, right-click the particular mining structure in the Object Explorer, and then click Browse. Microsoft has also included a set of data-mining web controls (and plug-in algorithms) as part of SSAS. You can use these to create custom applications for end users. If you choose to install the samples, you can find them at `C:\Program Files\Microsoft SQL Server\100\Samples\Analysis Services\DataMining`.

For example, the Microsoft Decision Trees algorithm includes two types of viewers: the Microsoft Tree Viewer and the Microsoft Content Viewer. The tree viewer itself has two different views of the mining-model output: Decision Tree view and Dependency Network view.

Figure 14–5 shows a portion of the Decision Tree view for the targeted mailing sample mining structure and the associated mining legend. It shows the most closely correlated information at the first level: in this case, Number Cars Owned. The depth of color of each node is a visual cue to the amount of association; darker colors indicate more association. Note that the mining legend reflects the exact number of cases (or rows) for the particular node of the model that is selected. It also shows the information via a probability column (percentage) and a histogram (graphical representation). In the diagram, the selected node is Number Cars Owned = 2 so the detailed (case) data in the mining legend reflects the selection.

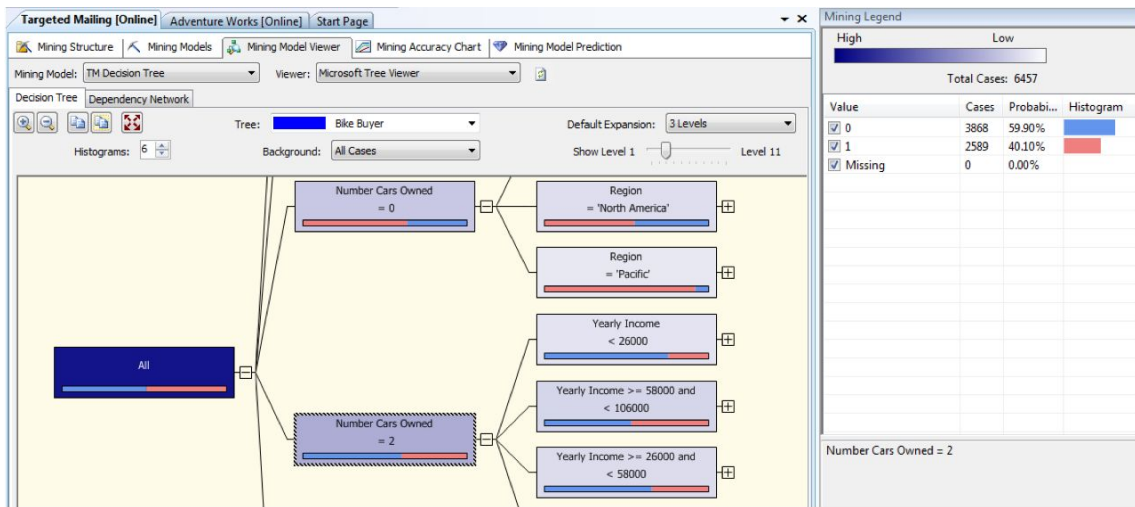


Figure 14–5. The Decision Tree view allows you to understand dependency relationships in your mining model.

The other type of view for the Microsoft Decision Trees algorithm is the Dependency Network view. This allows you to quickly visualize which data has the strongest correlation to a particular node. You can adjust the strength of association being shown by dragging the slider on the left of the diagram up or down.

Figure 14–6 shows the dependency network for the same mining structure you’ve been working with. Note that the four most correlated factors for bike purchasing are yearly income, number of cars owned, age, and region.

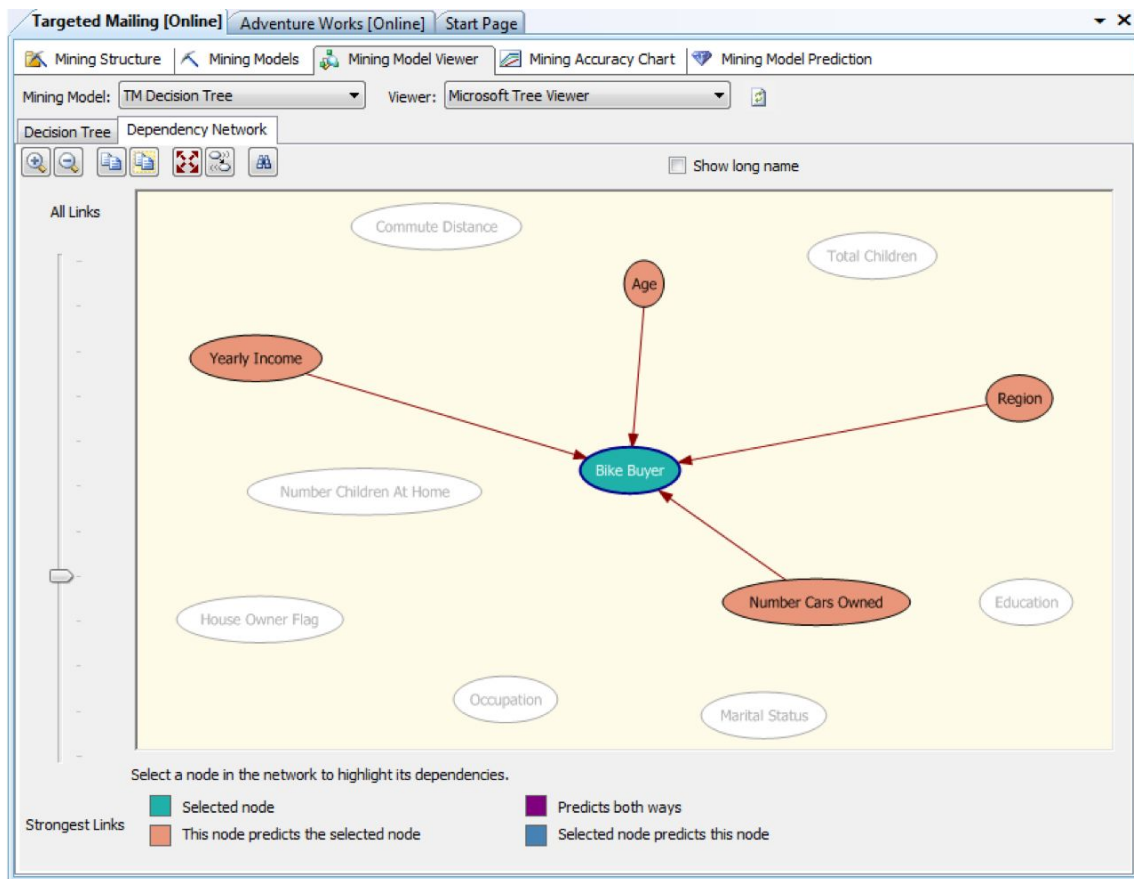


Figure 14–6. The Dependency Network view provides you with a visual representation of closely correlated factors. Note the legend at bottom left in the figure.

The Mining Model Viewer tab allows you to review the greatest level of detail in your mining model. It shows the processed results in rows and columns of data. For certain mining models, this viewer includes nested tables in the results as well.

As mentioned earlier, different algorithms have different types of associated views. For example, the Microsoft Naïve Bayes algorithm includes the following additional viewers: Attribute Profiles, Attribute Characteristics, and Attribute Discrimination. Figure 14–7 shows the Attribute Profiles view and includes the mining legend (which shows the color-coded buckets of data). This is showing the distribution of various attributes (age, commute distance, and so on) for the entire population (the complete data set), and then for the bike buyers—indicated by a 1 value—and the nonbike buyers—indicated by a 0 value.

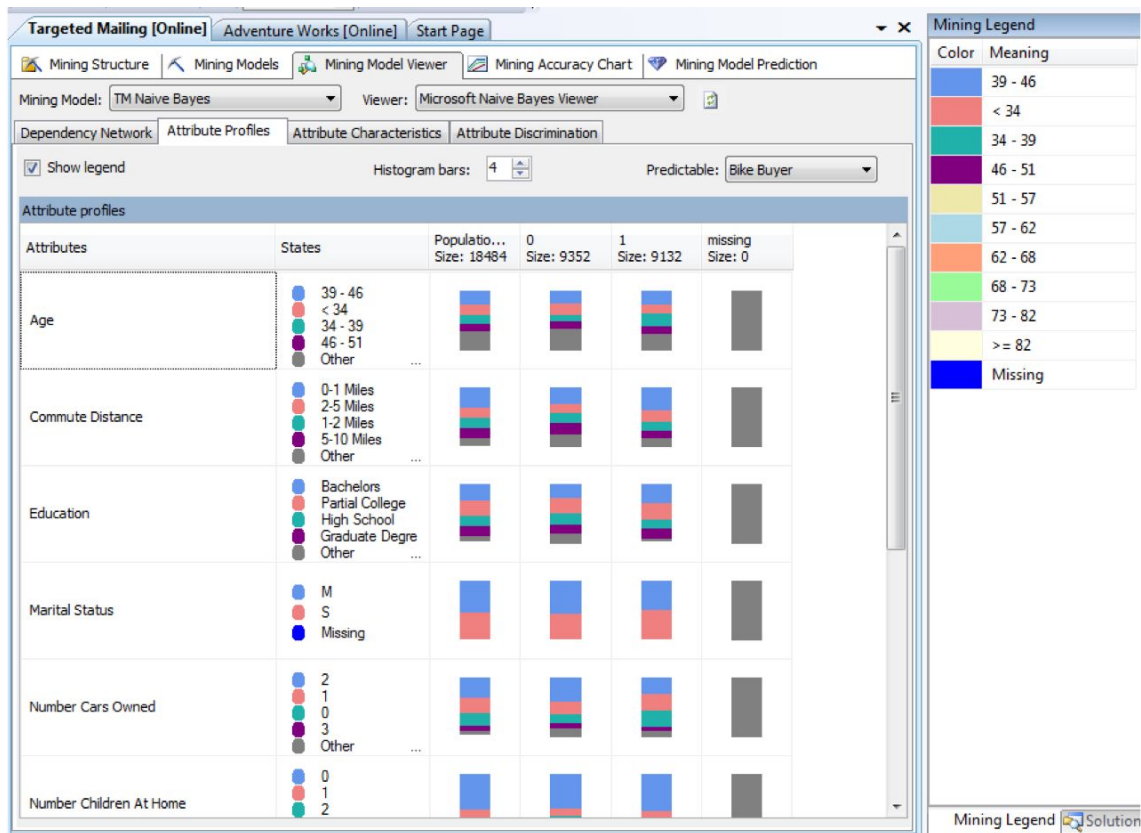


Figure 14–7. The Attribute Profiles view for a mining model using the Microsoft Naïve Bayes algorithm

Tip If you are new to data-mining concepts, a great way to better understand what the different included algorithms do is to look at the output from the included sample mining structures using each of the included views.

Mining Accuracy Chart Tab

The next tab in the BIDS designer is the Mining Accuracy Chart tab. Here you can validate (or compare) your model against some actual data to understand how accurate your model is. You do this by using a specific type of chart called a *lift chart*. This chart shows a perfect result and then compare the results of your particular mining model to that result.

You validate the various mining models inside a mining structure so you can understand which mining models will be most accurate (and, therefore, most useful) for understanding and impacting the particular business problems you want to forecast, predict, cluster, classify, and so on. Figure 14–8

shows the configuration of the Mining Accuracy Chart tab in BIDS, using the Specify Column Mapping dialog.

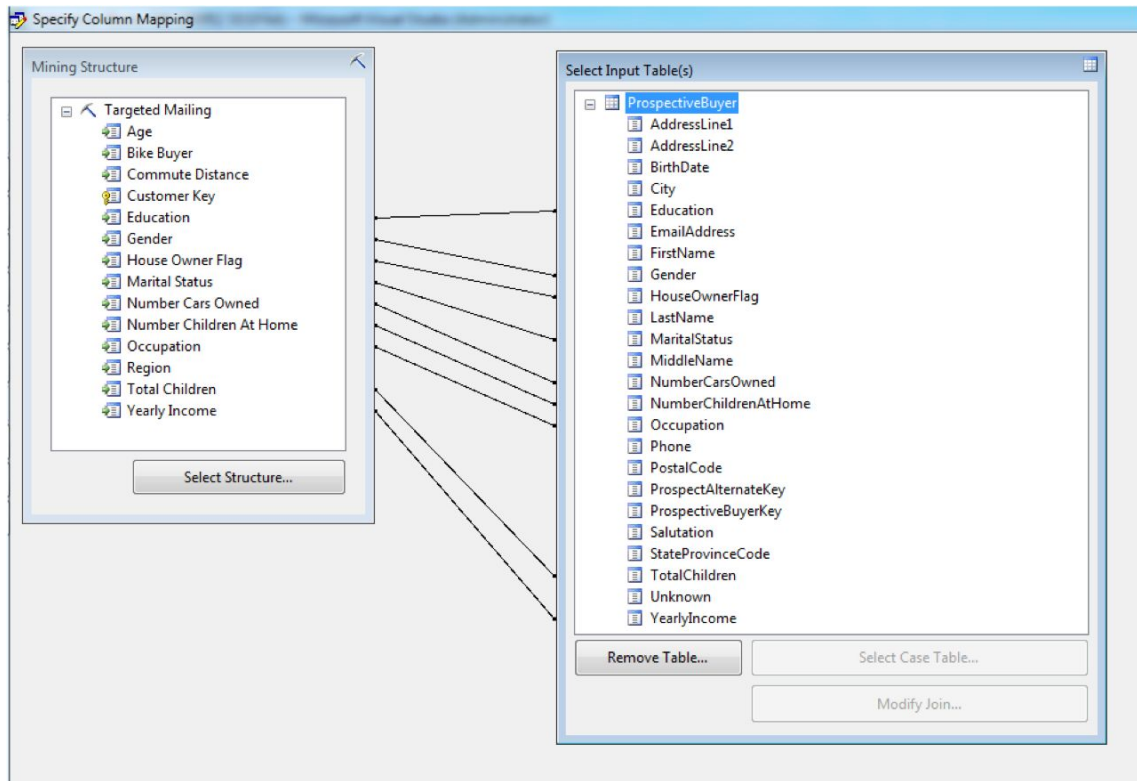


Figure 14–8. You associate a subset of data with the training (source) data, using column mapping.

To further explain the outputs of the Mining Accuracy Chart tab, you need to understand a bit more about lift charts and profit charts. A *lift chart* compares the accuracy of the predictions of each model included in your structure to an average guess and a perfect prediction. This guess is shown in the middle of the chart, and the perfect value is shown as a line at the top of the chart. *Lift* is measured by how much your model results improve (or lifts) the predictive values above that average guess. A *profit chart* displays the theoretical increase in profit that is associated with using each model.

Mining Model Prediction Tab

The next tab in the BIDS designer is the Mining Model Prediction tab. Here you can create predictions based on the mining models that a mining structure contains. What you are actually doing is writing a query using the native mining-model language—Data Mining Extensions (DMX)—using the designer interface. You can use DMX to create the structure of new data-mining models, to train (or populate) your models, and to browse, manage, and predict against them. DMX contains Data Definition Language (DDL) statements, such as `CREATE MINING STRUCTURE`; Data Manipulation Language (DML)

statements, such as `SELECT FROM <model>;` and functions and operators, such as `Predict` and `PredictAssociation`. We will look in more detail at DMX sample statements later in this chapter.

Figure 14–9 shows the interface in BIDS for creating a data-mining prediction query. This example is working with the DMX `PredictProbability` function to determine the probability of an individual (case) becoming a bike buyer.

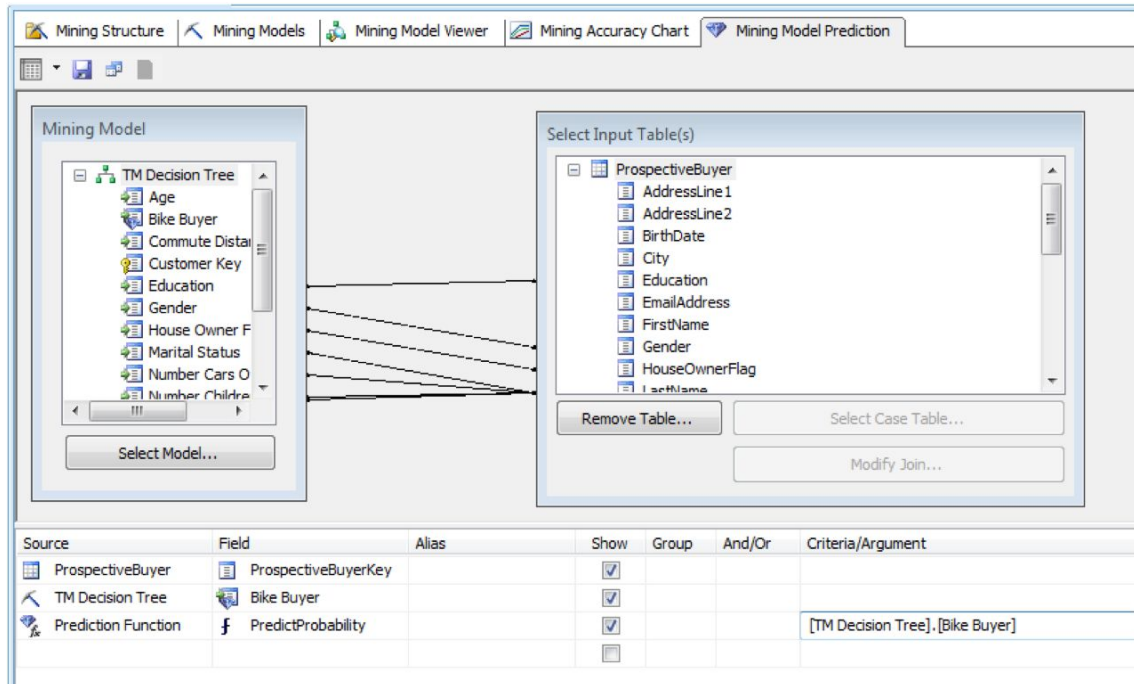


Figure 14–9. The BIDS interface for generating a mining-model prediction query allows you to easily create predictive queries.

Understanding and Using the Included Data Mining Algorithms

SQL Server ships with nine built-in data-mining algorithms. Before we start reviewing the individual algorithms, there's a concept that will help you understand how to select the mining algorithm that best matches your business needs: *supervision*. Your options are to use a supervised or an unsupervised algorithm. Here is the difference.

A *supervised mining models* require you to select both input and predictable columns. This type is used when you not only know the values (or columns) that should serve as inputs but also when you have determined which value you want to predict.

An *unsupervised mining models* require you only to select input columns. This type is used earlier in your project when you are just trying to understand the data you are working with, rather than to predict a specific outcome.

■ **Note** An *input column* is any column from source data that should be considered when executing the mining algorithm. A *predictable column* is any column for which you want the algorithm to predict values. Not all algorithms support prediction (some support only clustering or grouping data together).

When you build your models, you're presented with an error dialog box if you don't configure your model per the supervision requirements for the particular algorithm you've selected. The unsupervised algorithms are Microsoft Clustering, Microsoft Linear or Logistical Regression, Microsoft Sequence Clustering, and Microsoft Time Series. The supervised algorithms are Microsoft Association Rules, Microsoft Decision Trees, Microsoft Naïve Bayes, and Microsoft Neural Networks.

The Nine Algorithms

Following is a list of all the algorithms in descending order of most common use. The list also includes a brief discussion of some of the more important configurable attributes for each algorithm.

- *Microsoft Naïve Bayes*: One of the simplest algorithms available to you in the SSAS toolkit, so it is often used as a starting point to help you understand basic groupings in your data. It's called *naïve* because no one attribute has any higher significance than another. Also, only discrete (distinct and not fractional) content types can be evaluated. This model type is most often used to view and understand data better; it's not used as commonly to predict because of its naïve nature. There are no configurable properties available with this algorithm when you click the algorithm parameters property Build button in the Solution Explorer in BIDS. The most commonly used viewer in BIDS for this type of model is the dependency network.

■ **Tip** The Algorithm Parameters dialog box in Solution Explorer in BIDS only shows a partial list of the configurable properties for each algorithm. For several algorithms, it shows none at all. If you want to add a configurable property and a value, click the Add button at the bottom of the dialog box. If you search on the name of the particular algorithm in BOL, you can review the complete list of configurable properties for each algorithm.

- *Microsoft Decision Trees*: Probably the most commonly used algorithm, due to its flexibility. It works with discrete and continuous attributes (numbers that represent some unit of measure, which can include fractions). The richness of the included viewers makes it quite easy to understand the algorithm's output. This algorithm is used to both view and to predict. There are four configurable properties with this algorithm. The most important property is `COMPLEXITY_PENALTY`. By adjusting this number (usually downward), you can decrease the complexity of your model by reducing the number of inputs to be considered and literally reducing the size of your decision tree—that is, the number of nodes in the result set. The most commonly used view for this algorithm is the Decision Tree view. It allows you to view the nodes representing the most closely aligned factors predicting the value you select, and includes supporting information that shows detail about the values and the number of cases for each node.
- *Microsoft Time Series*: Used to impact a common business problem: accurate forecasting. This algorithm is often used to predict future values, such as rates of sale of a particular product. The configurable properties include the interesting `PERIODICITY_HINT` property. Supplying a value for this setting allows you to nudge the algorithm into a better understanding of the way the data is distributed across time. For example, if sales vary by year, and the unit of measurement in the series is quarters, then the periodicity is 4. The Time Series Chart view helps you understand the output of your model by showing the predicted values over the configured time series in an easy-to-understand graphical format. You can also configure the number of prediction steps (the number of data points along the time axis) and whether you want to show deviations (outliers) by using the controls on the viewer design surface. This is shown in Figure 14–10.

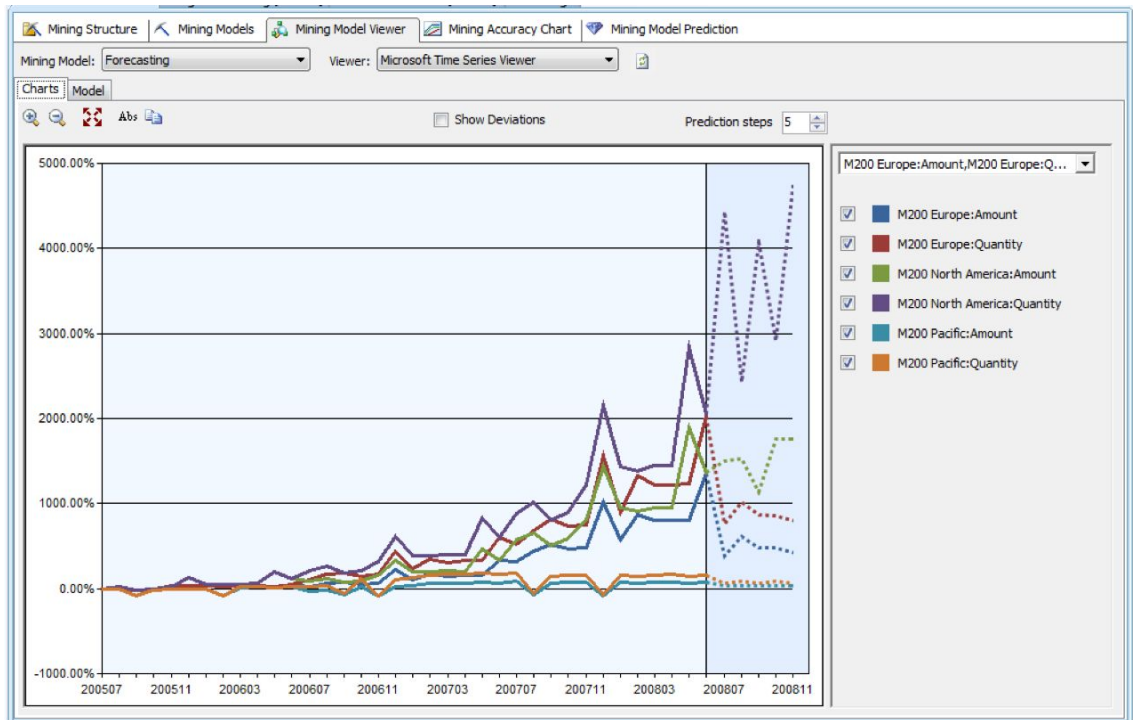


Figure 14–10. The Time Series chart allows you to see the selected, predicted values over the time series.

- Microsoft Clustering:** Separates your data into intelligent groupings by iteratively grouping cases into clusters that contain similar characteristics. This algorithm can help you bucketize types of customers, for example. You can configure CLUSTERING_METHOD using the following properties available for this algorithm: Scalable Expectation Maximization (EM), Nonscalable (vanilla) EM, Scalable K-means, and Nonscalable K-means. The default is scalable EM. K-type clustering is considered *hard* clustering in that it creates buckets or grouping and then assigns your data into only one bucket—there is no overlap. EM clustering takes the opposite approach and allows overlaps; this type is sometimes called *soft* clustering. After you've created your model, it's important to use the associated views to better understand the clusters that have been created. You can also rename the clusters shown in the Cluster Diagram view as you work with the other tabs to better understand the traits associated with a particular cluster.

- Microsoft Sequence Clustering:** Does the same thing as clustering, with one important addition—it monitors the states between values. You can use this to get a sense of page sequences customers use on your web site, for example. The State Transitions view for this algorithm is particularly interesting. Using that tab in BIDS, you can look at the state transitions for any selected cluster. Each square (node) represents a state of the model, such as “water bottle.” Lines represent the transitions between states, and each node is based on the probability of a transition. The background color represents the frequency of the node in the cluster. The number displayed next to the node represents the probability of affecting the associated node. This view is shown in Figure 14–11.

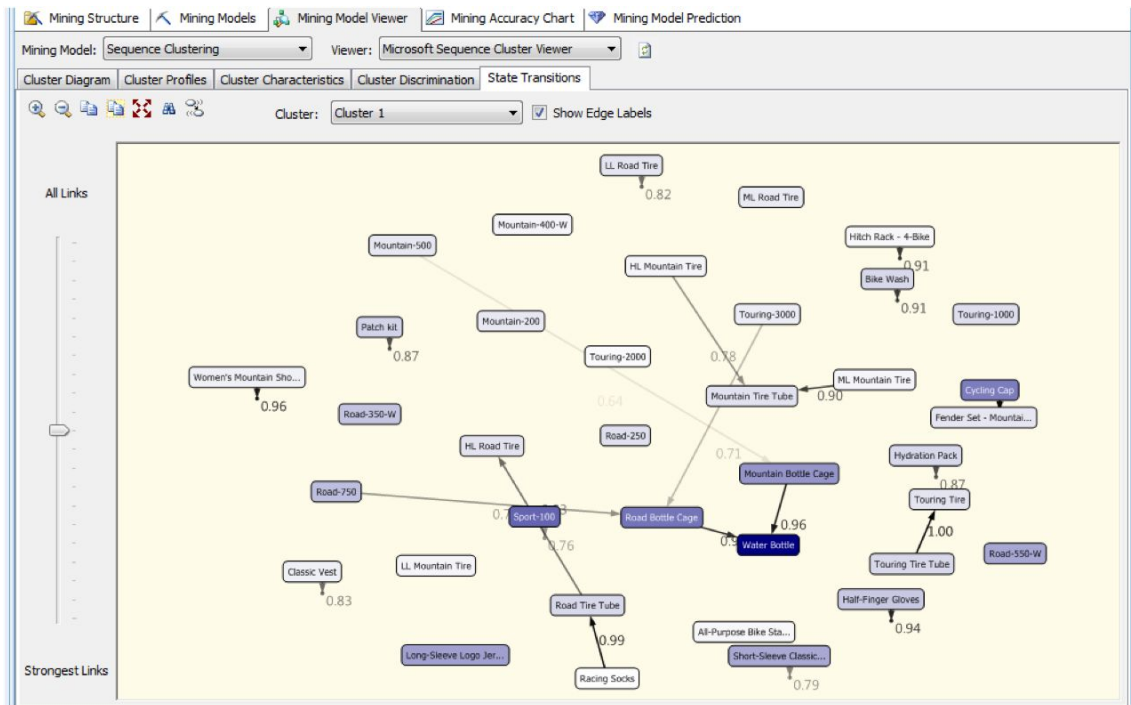


Figure 14–11. The State Transitions view allows you to visualize the transitions between states for each cluster.

The most interesting configurable property for this algorithm is the `CLUSTER_COUNT`. This allows you to set the number of clusters that the algorithm builds. As with Microsoft Clustering, by adjusting the number of clusters, you can gain a better perspective on the data. Most often, you will reduce the number of clusters so you can focus on the most important results-generating clusters for your particular business scenario. The default number of clusters generated is ten.

- Microsoft Association Rules:** Produces *itemsets*—groups of related items from the source attribute columns. The results of this algorithm are often called *market-basket* analysis. There are four configurable properties for Microsoft Association Rules. Included in the configurable properties is the ability for you to adjust the maximum size of the discovered itemsets (`MAXIMUM_ITEMSET_SIZE`). This property is set to a value of 3 by default, which means the three items that are sold most often together are shown as itemsets in the results views.

The most interesting view for the Association Rules algorithm is the Rules view. It shows whatever subset groupings of itemsets per your configuration parameters; for example, if you've stayed with the default value of 3 for the `MAXIMUM_ITEMSET_SIZE` property, then you see groupings of up to three related items in the associated mining-model views. Note that you can adjust the view by setting or adjusting filters, min/max probability settings, and other properties. This view is shown in Figure 14–12.

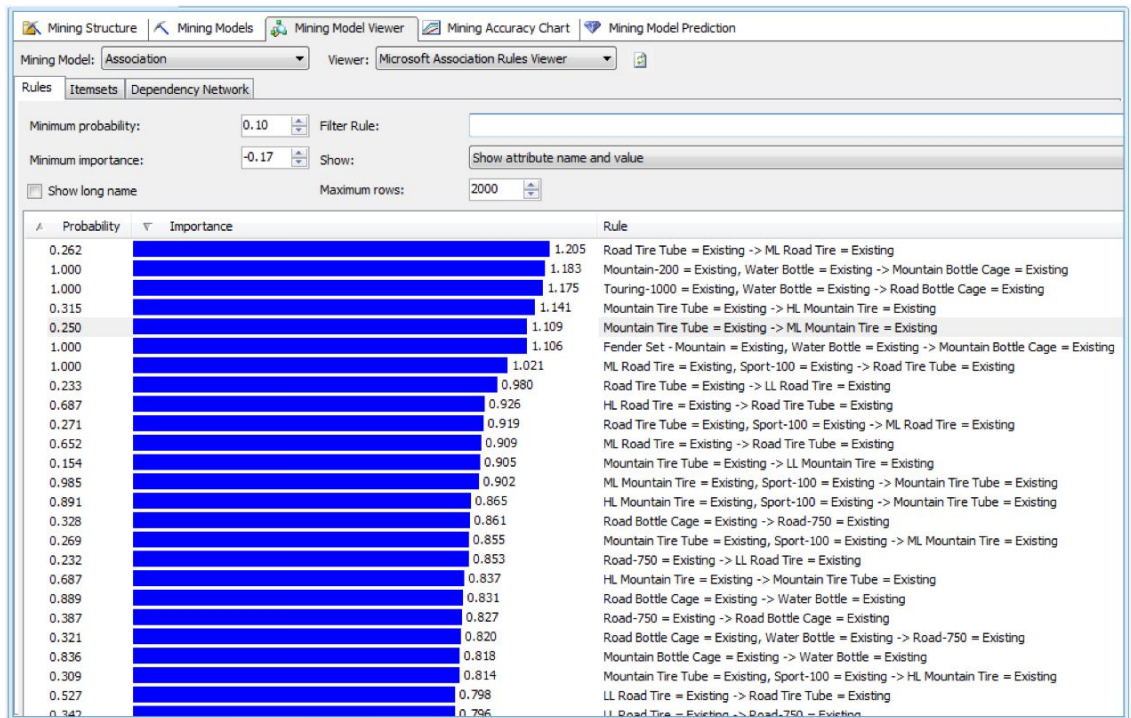


Figure 14–12. The Rules view for the Microsoft Association Rules algorithm

- Microsoft Neural Network:** By far the most powerful and complex algorithm, which is often used when other algorithms fail to produce meaningful results. Using the Microsoft Neural Network algorithm against large data sources should always be well-tested using near production-level loads. This is because of the amount of overhead needed to process these types of models. The algorithm itself uses a very complex network called the multilayer perceptron network. This network uses three types of objects: input, hidden, and output neurons. It has no configurable parameters in the associated dialog box in the BIDS GUI. However, you can add and configure parameters such as `MAXIMUM_INPUT_ATTRIBUTES` and `MAXIMUM_OUTPUT_ATTRIBUTES`, both of which have a default value of 255. You may choose to adjust either of these to improve both the performance and the usability of the results of this algorithm.

For more information about the details of how this algorithm processes data and the possible configurable parameter names and values that you may add to it, see the BOL topic “Microsoft Neural Network Algorithm (SSAS).” Figure 14–13 shows one of the built-in views for this algorithm, displaying the Yearly Income, Total Children, Region, and Occupation attributes, and how they contribute to predicting female bike buyers.

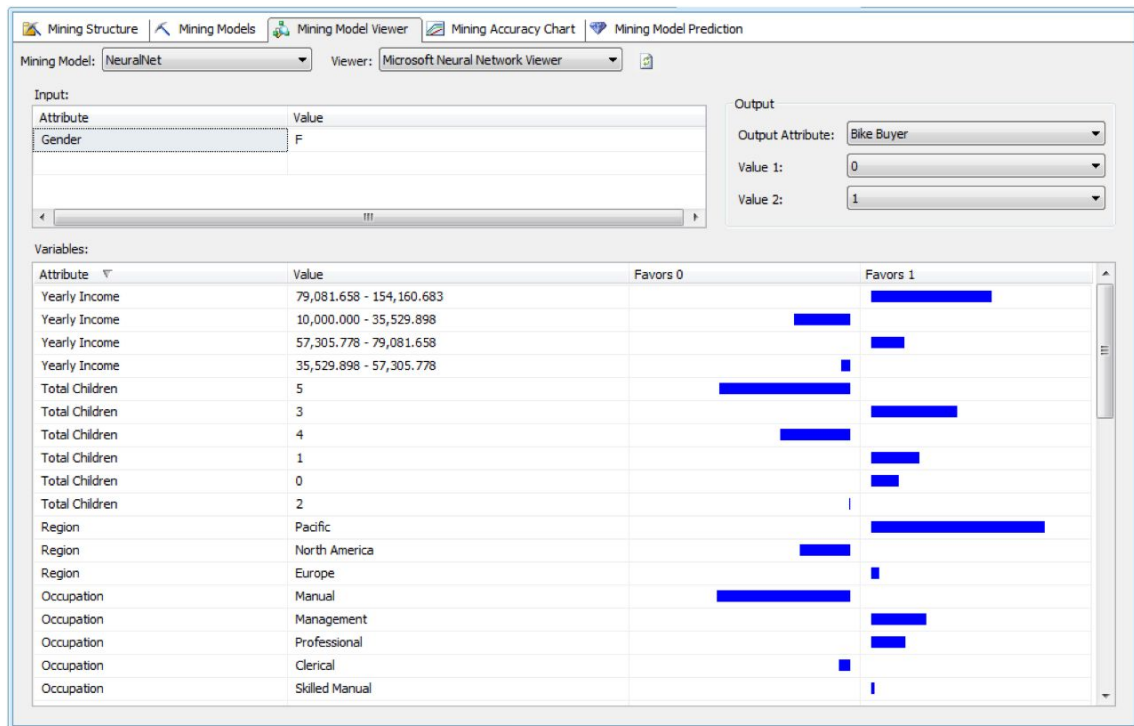


Figure 14–13. The built-in Mining Model view for the Microsoft Neural Network algorithm

- *Microsoft Linear Regression*: A variation of the Microsoft Decision Trees algorithm that works like classic linear regression; that is, it fits the best possible straight line through a series of points. It has no included configurable parameters.
- *Microsoft Logistic Regression*: A variant of the Microsoft Neural Network algorithm with a `HIDDEN_NODE_RATIO` parameter set to 0, which makes its output more like a variant of linear regression. One example is when the dependent variable is a dichotomy, such as success/failure. It has no included configurable parameters.

Now that we've reviewed the nine mining algorithms, your next step is to consider the complete set of information that you'll need to use to create your mining model and how exactly to implement data-mining structures and models in BIDS.

The Data Mining Wizard

In this section, you explore the Data Mining Wizard. Using this wizard, you define new mining structures by supplying data from relational tables or an OLAP cube. The wizard helps you build your model by analyzing your data and recommending how each data point may be used.

To start the wizard, right-click the Mining Structures folder in Solution Explorer in BIDS, and select New Mining Structure. As you work with the wizard's dialog boxes to create a structure, you need to consider these questions:

- What type of data can you include: relational (from a DSV) or multidimensional (from a cube in your project)? For the purpose of creating a mining structure, you can use either type of data. The wizard supports both types equally well. Your selection of data source should be based on where the data that you need is stored.
- For relational sources, which table is the *case table* (describes the main entity, such as customers)? Which is the *nested table* (has information related to each case or entity; in a many relationship, for example, sales transactions for each customer)? Do the case and any nested tables have columns that form a relationship between rows in the case (one: primary key) table and rows in the nested (many: foreign key) table(s)? What other columns will you include from both tables?
- For OLAP sources, what dimensions, attributes, and facts will you include?
- Which algorithm will you start with? Remember, you can easily add algorithms to your mining structure after you've completed the wizard.
- For the included columns, which are key, input, or predictable? You can select more than one type of column (input, predictable, and so on) for each column. The Data Mining Wizard has built-in intelligence to help you. You can use the Suggest button at the bottom of the dialog box; the result is shown via a dialog box (see Figure 14–14).

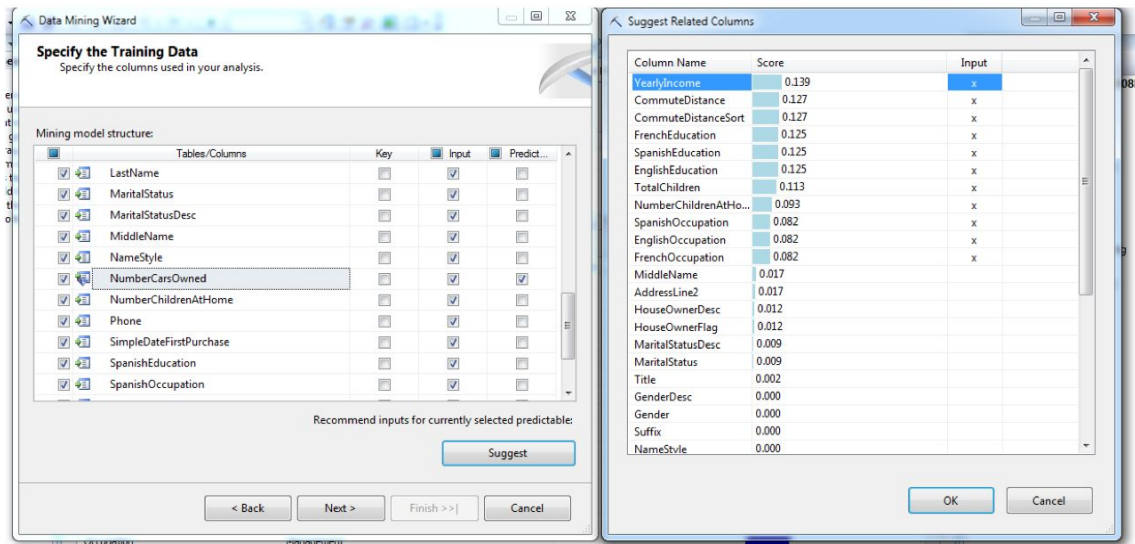


Figure 14-14. The Data Mining Wizard can help you select input columns for your mining structure by showing you which columns are most related to your predicted attribute after you click the Suggest button in the wizard's dialog box.

- What are the content and datatypes of the columns?

In the next step of the wizard, you are presented with a dialog box that shows you the content type and datatype of the involved columns. You can make adjustments as needed in this dialog box. There is also a handy Detect button at lower right that you can use to help you configure the appropriate content type (continuous or discrete) for a numeric column (see Figure 14-15). These two concepts are discussed in greater detail in the next section of this chapter.

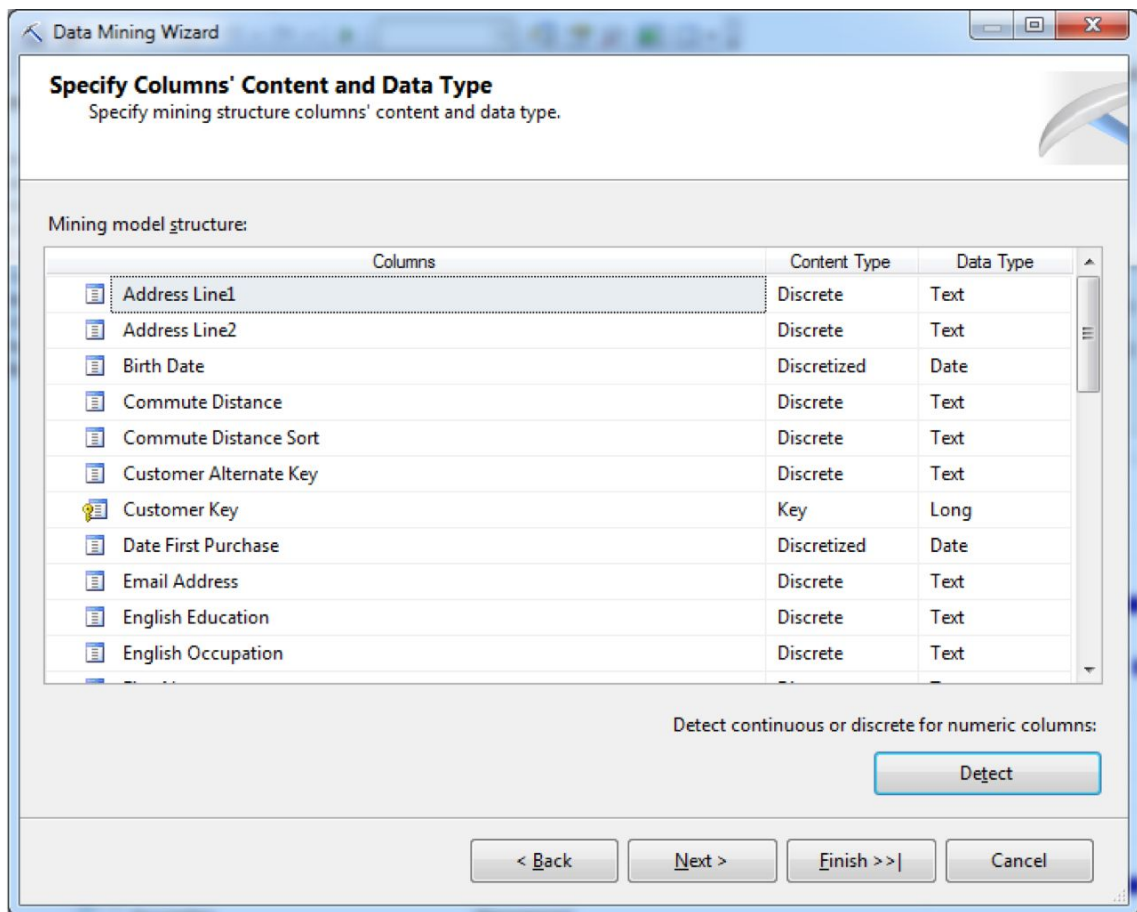


Figure 14–15. The Detect button can help you determine the appropriate content type setting for numeric columns: continuous or discrete.

Content and Datatypes

SSAS data-mining structures use data and content types that are specific to the Microsoft implementation of data mining. It's important that you understand these types when you build your mining structures. Also, certain algorithms only support certain content types. A *content type* is an additional attribute that the mining model uses to understand the behavior of the data; an example is *cyclical*. Marking a source column as a cyclical content type tells the mining algorithm that the order of the data is particular, important, and repetitive, or has a cycle to it, for example, the month numbers of more than one year in a time table.

A *datatype* is the same as what you already understand datatype to mean from relational database modeling, that is, integer, string, and so on. The difference here is that the datatypes are based on C++, rather than the typical relational database datatypes that you are probably more familiar with. The rule of thumb is for you to determine the datatype first and then verify (and sometimes adjust) the

appropriate content type in your model. Remember that certain algorithms only support certain content types; for example, Naïve Bayes does *not* support continuous content types.

Here's a list of the content type attributes:

- *Discrete*: This column (or attribute type) contains values that are distinct—for example, number of children. Another way to think about this is that there are no fractional values. It is worth noting that marking a column as discrete does *not* indicate that the order (or sequence) of the information is important. You can use any datatype with this content type.
- *Continuous*: This column has values that are a set of numbers representing some unit of measurement—for example, outstanding loan amount. These values can be fractional. You can use the date, double, and long datatypes with this content type.
- *Discretized*: This column has continuous values that are grouped into buckets. Each bucket is considered to have a specific order and to contain discrete values. You can use dates, doubles, longs, and text datatypes with the discretized content type.

■ **Note** You can adjust `DiscretizationMethod` (which determines how the data will be grouped: `AUTOMATIC` | `CLUSTERS` [for numerics only; works well for any type of distribution curve] or `EQUAL_AREAS` [for strings; works best for standard distribution curves]) and `DiscretizationBucketCount` (which specifies the number of discrete buckets into which the data in this column should be divided) if you mark your column as `Discretized`.

- *Key*: This column is used as a unique identifier for a row. You can use date, doubles, longs, or text for this.
- *Key Sequence*: This column is a type of a key; the sequence of key values is important to your model. You can use doubles, longs, text, or dates with this content type. By using this content type, you are identifying the importance of the sequence (order) of the key values, in addition to noting that the values are identifiers or keys.
- *Key Time*: This column, similar to a key sequence, is a type of key where the sequence of values is important. Additionally, by marking your column with this content type, you are indicating to your mining model that the key values run on a time scale.
- *Ordered*: This column contains data in a specific order that is important for your mining model. Also, when you mark a column with the `Ordered` content type, SSAS considers that all data contained is discrete. You can use any datatype with this content type.
- *Cyclical*: This column has data that is ordered and represents a set that cycles (or repeats). It is often used with time values (months of year, for example). Data marked as `Cyclical` is considered both ordered and discrete. You can use any datatype with this content type.

For your reference, we’ve pivoted the information presented in the preceding list in Table 11-1 so that you can see which datatypes are supported by which content types. Understanding this concept is very important to successful model building.

Table 11-1. List of Datatypes and Supported Content Types for Each One

| Datatype | Content Types Supported |
|----------|---|
| Text | Discrete, discretized, or sequence |
| Long | Continuous, cyclical, discrete, discretized, key sequence, key time, ordered, sequence, or time |
| Boolean | Discrete |
| Double | Cyclical, discrete, discretized, key sequence, key time, ordered, sequence, or time |
| Date | Continuous, discrete, discretized, or key time |

As you complete the wizard, you’re asked to name your mining structure. You have the option to allow drillthrough for your mining structure here as well. Drillthrough in this context functions similarly to drillthrough in a SSAS cube; that is, it gives the end users with appropriate permissions the ability to right-click a data-mining model to see a list of the source data columns that lead to the result that they are viewing. Remember that any columns that are included in drillthrough must be included in the particular model; that is, end users cannot drill through to data that is part of the source start schema but has not been included in the SSAS database.

If you select From Existing Cube as your data source for your mining model, you’re presented with one additional dialog box before you complete the wizard. This dialog box allows you to define a particular slice of the cube for your mining structure. Figure 14–16 shows an example of this, using the slice “show only customers who are home owners.”

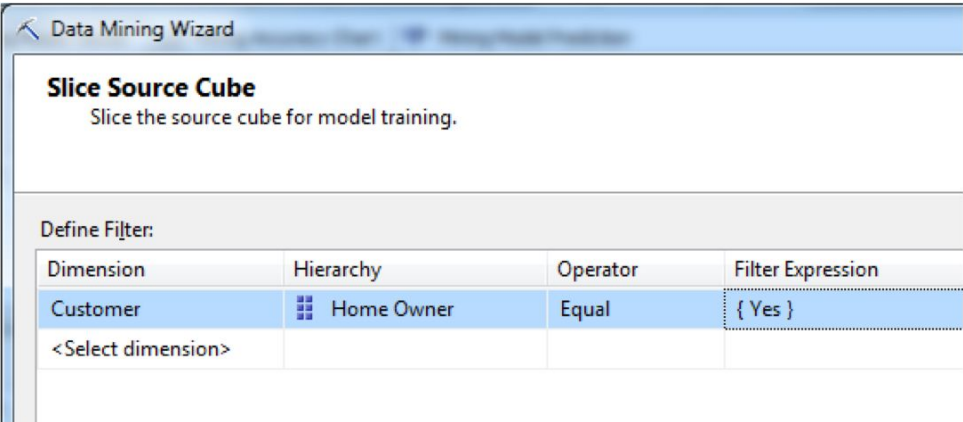


Figure 14–16. The Data Mining Wizard allows you to define slices of the source cube when you select From Existing Cube as the data source for your mining structure.

■ **Tip** If you are creating a model based on an existing cube that contains a time series, then you need to slice the time dimension to remove all future members, assuming those members have been loaded into your cube already.

Another difference in the wizard, if you base your mining model on an existing cube (and you use any of these three algorithms: Microsoft Clustering, Microsoft Decision Trees, or Microsoft Association Rules), is that in the final dialog box, you are also asked whether you want to create a new dimension in your existing cube or create a new cube with a data-mining dimension in it. This is shown in Figure 14–17.

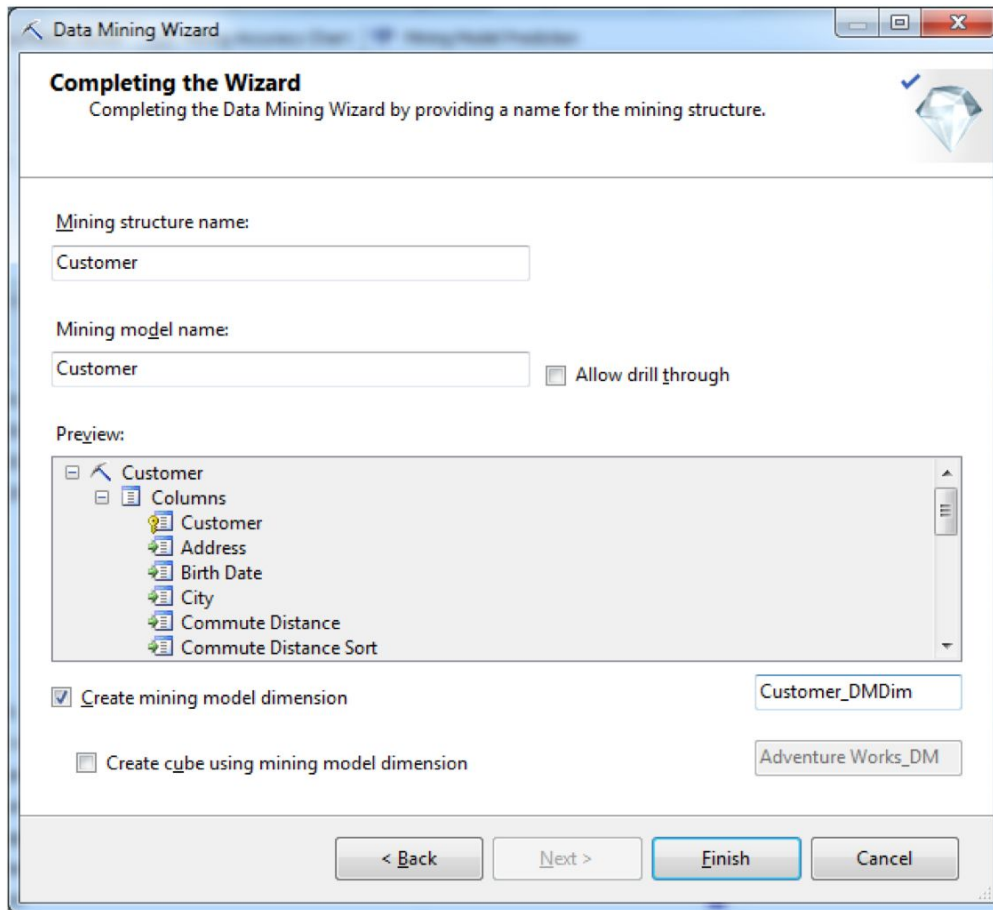


Figure 14–17. The Data Mining Wizard allows you to create a new dimension in your existing cube using the information in your mining model, or you can create an entirely new cube containing the new dimension.

One interesting aspect of creating a new dimension, whether in the existing cube or in a new cube, is that a new DSV is created representing the information in the model. Unlike most typical DSVs, this one cannot be edited, nor can the source data (the model output) be viewed via the DSV interface.

Processing Mining Models

After you complete the design of your mining structure, you must process it. You can process some types of included mining models individually, or you can process all models included in the particular mining structure that you are working with.

Processing Methods

To begin, you should understand the different methods at your disposal. Processing methods for mining *structures* are as follows:

- *Process Full*: Drops all data and metadata and completely reprocesses the selected object.
- *Default*: Detects the selected object's current state and then determines the appropriate processing method.
- *Process Structure*: Populates only the mining structure but not the mining models with source data.
- *Process Clear Structure*: Removes all training data from the selected object.
- *Unprocess*: Deletes the data in the object selected and any lower-level associated objects.

Process methods for mining *models* are as follows:

- *Process Full*: Drops all data and metadata and completely reprocesses the selected object.
- *Default*: Detects the selected object's current state and then determines the appropriate processing method.
- *Process Unprocess*: Deletes the data in the object selected and any lower-level associated objects. After the data is deleted, it is not reloaded.

As with cube processing, you can configure error handling during processing by clicking the Change Settings button in the Process Model (or Structure) tab dialog box.

Figure 14–18 shows you the detailed output window that is generated in BIDS when you choose to process your mining model.

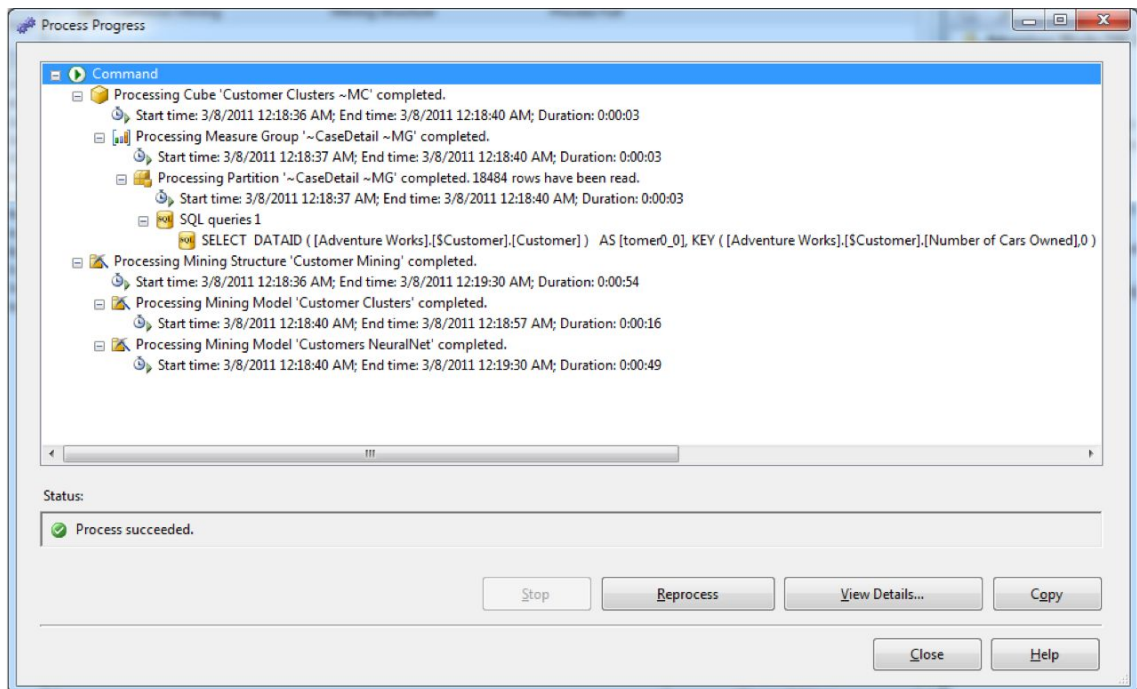


Figure 14–18. The Process Progress dialog box for data-mining structures shows a detailed record of all activities performed during mining-structure processing.

After you move to production, you should automate the processing of your mining structures. You can easily do this by creating a SQL Server Integration Services (SSIS) package and including the Analysis Services Processing Task. You configure the task to perform the type of processing that your business requirements require for the particular mining structure. This task is encapsulating the XML for Analysis (XMLA) Process command, so if you wanted to script this process using only XMLA and not SSIS, you could do that as well. In addition to automating model-structure processing, there are a couple of other areas of integration between data mining and SSIS. We'll review those in the next section.

SSIS and Data Mining

Data mining by DMX query is supported in SSIS. You have one control-flow SSIS task and one data-flow transformation to select from. The Data Mining Query Task is available in the Control Flow toolbox. The editor for this task is shown in Figure 14–19.

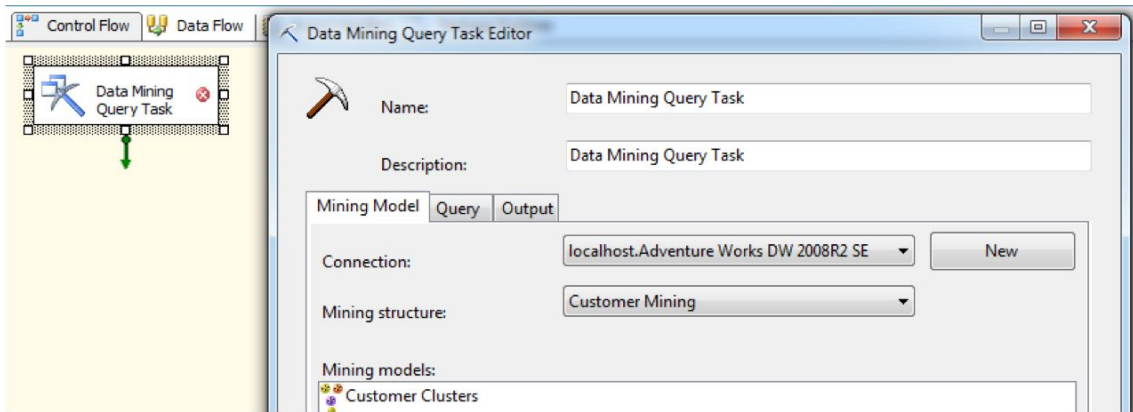


Figure 14–19. The new Data Mining Query Task allows you to associate a data-mining query with the control flow of an SSIS package.

The Data Mining Query Transformation is available in the Data Flow Transformation toolbox. As with the control-flow task, the data-flow transformation allows you to associate a data-mining query with an SSIS package. One difference between the two objects is, of course, where they are used in your package. One is for control flow (which connects various activities together, such as T-SQL queries, FTP, WMI, and so on), and the other is for data flow (which impacts a specific set of data flowing through the package by extracting, transforming, and then loading the transformed data to one or more destinations).

An example of where the Data Mining Query Transformation can be used is a situation in which you have a large amount of unclean data. You can use this transformation to help determine possible values for partially dirty data by splitting this data into clusters. The dialog box for this transformation is shown in Figure 14–20. Note that both of these objects are available only with the Enterprise Edition of SQL Server.

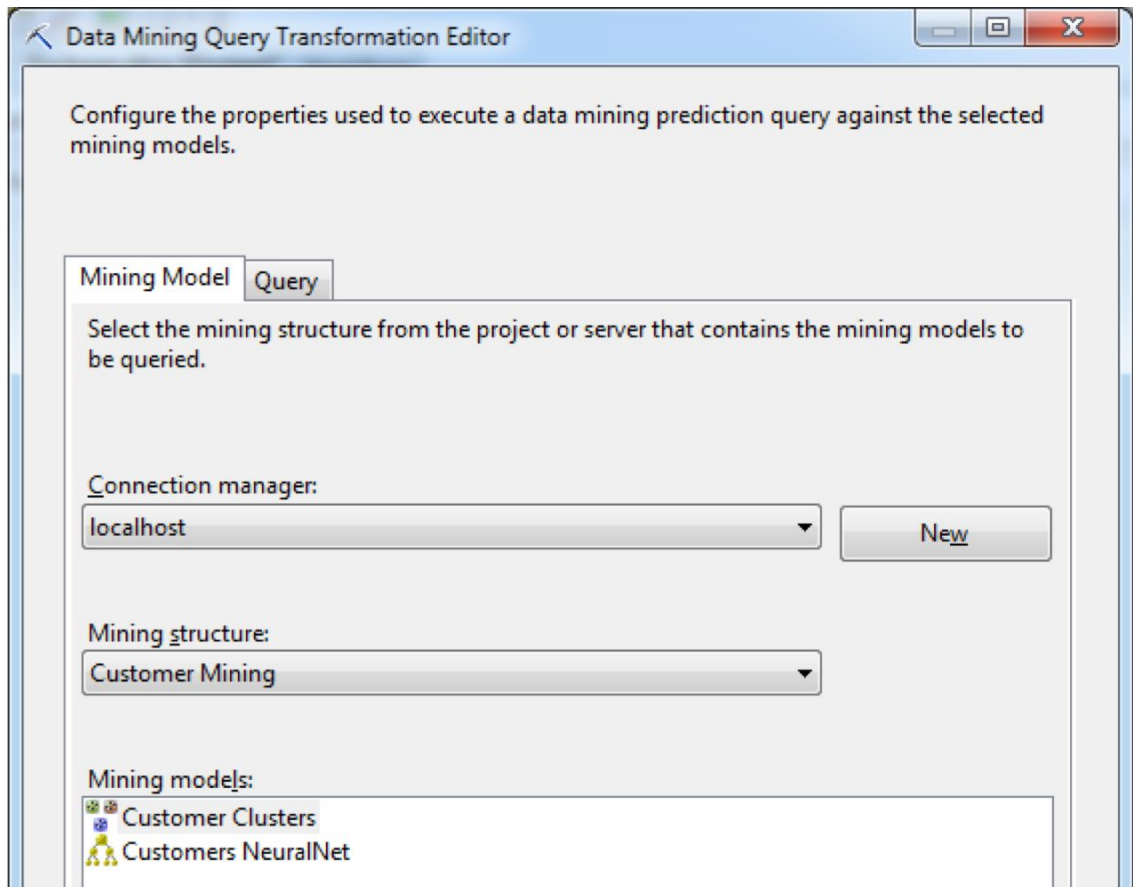


Figure 14–20. The new Data Mining Query Transformation allows you to associate a data-mining query with the data flow of an SSIS package.

Working with the DMX Language

If your business requirements call for custom client development using the data-mining API, you can begin to familiarize yourself with the DMX language by reviewing the supplied templates in SSMS. As with creating an SSAS query, you first connect to SSAS in SSMS, display the Template Explorer, and then click the DMX node to view and work with the included templates. Figure 14–21 shows the included DMX templates.

They include Model Content (which allows simple querying of model data and metadata, such as return model attributes), Model Management (which allows for administration, such as create, rename, and export model), and Prediction Queries (which allows use of model content in conjunction with other data using the OPENQUERY operator).

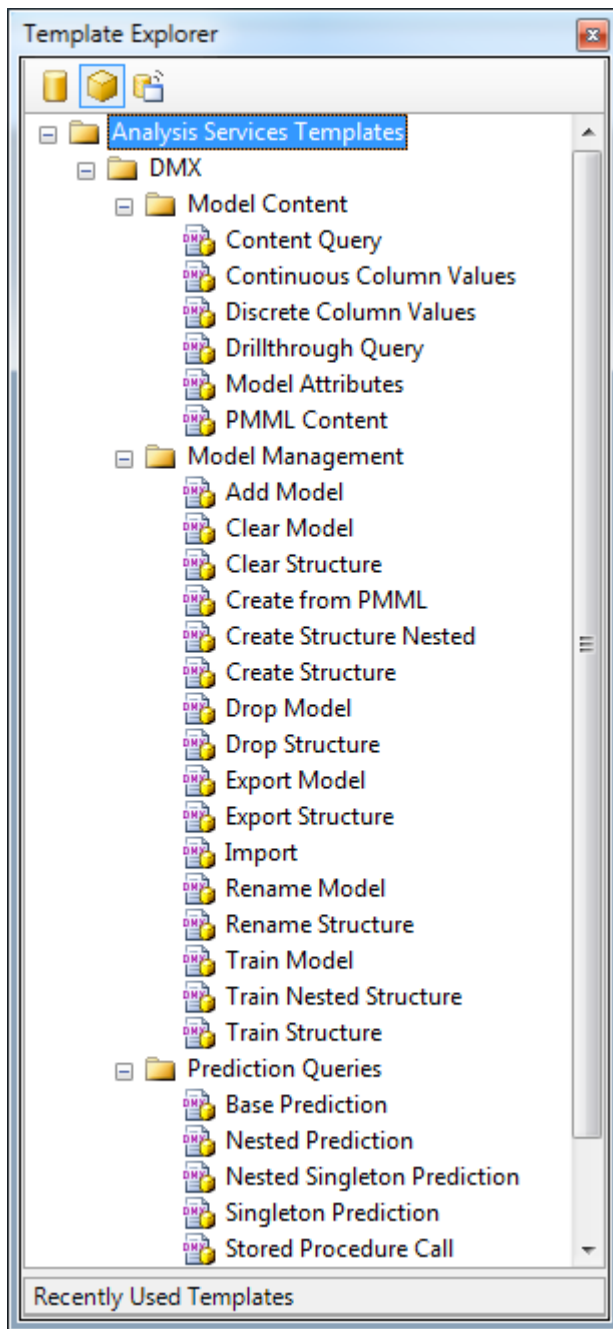


Figure 14–21. The Template Explorer in SSMS provides you with three different types of DMX queries.

As with Multidimensional Expressions (MDX), in our experience, you'll seldom have to manually write a large number of DMX queries in production BI solutions. One obvious exception to this general guideline is, of course, if your BI project work is primarily focused on implementing data mining. Microsoft has provided you with a large number of tools, templates, and wizards so that you can implement DMX queries without having to take the time to master the language from scratch. Because of this, we'll simply review the basic DMX syntax at this time.

SSMS includes a DMX query tool. You access this tool in a fashion similar to the way you worked with the MDX query tool: you open SSMS and click the Analysis Services DMX Query button on the toolbar in SSMS to open a DMX query window. Figure 14–22 shows the results of a simple DMX query in SSMS.

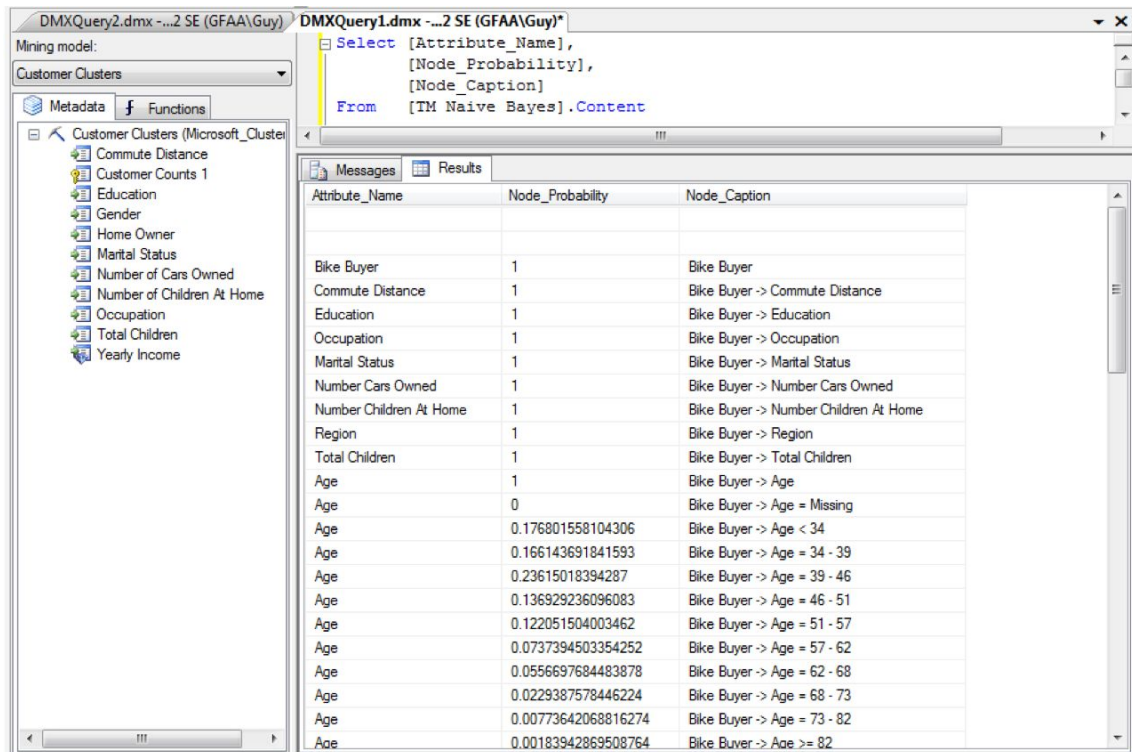


Figure 14–22. You can write and execute custom DMX queries in SSMS.

The query shown allows you to view metadata about a particular mining model. I built this query by customizing one of the included DMX templates. These templates are a very good starting location for learning DMX. As with any query language, if you are going to do extensive DMX work, you should probably pick up a reference book on it. You should also refer to the BOL topic “Data Mining Extensions (DMX) Reference.”

■ **Tip** To better understand what type of DMX query you might choose to implement using either of the SSIS objects, you may want to explore the DMX language in a bit more detail. A great way to learn any new language is to read queries generated by client tools before you begin to attempt to write queries. For data mining, you can do this by capturing queries that data-mining client tools (including the BIDS viewers) generate using the SQL Server capture tool—SQL Server Profiler. SSAS DMX queries can be captured using tracing with Profiler. DMX queries are noted in the Event Subclass column (type = 1) of Profiler traces.

Summary

One of the most important considerations when determining whether (and how) to implement SSAS data mining in your BI solution is the selection of end user or client tools. In the past, the lack of available client tools has definitely hurt the adoption of data mining. However, that void has been filled as of late. As of this writing, you have a few choices: you can write your own client tools, you can purchase third-party vendor client tools, or you can use Excel.

Are you now intrigued about the possibilities of adding data mining to your BI solution? We hope so! We covered data-mining concepts, conducted a brief tour of the BIDS data-mining UI using one of the AdventureWorks samples, and then went on to cover building data-mining structures in more detail. We discussed each of the supplied algorithms and looked at their content, datatypes, and other key properties. We then discussed the practical matters of processing your mining structures and took a quick look at SSIS integration. We ended our tour of SSAS data mining by looking at the basic syntax of DMX.



The HIERARCHYID Datatype

Microsoft introduced the HIERARCHYID datatype as part of SQL Server 2008. A HIERARCHYID instance represents a hierarchy tree structure. Each node of a hierarchy tree can be manipulated and retrieved from the HIERARCHYID instance. When using HIERARCHYID, access is provided via a managed set of Common Language Runtime (CLR) data type methods that support creating, reading, updating, and deleting (CRUD) nodes in a hierarchy. An employee dimension, usually modeled as a self-referencing parent-child dimension, is a good candidate for HIERARCHYID.

In the following sections, you will create an employee table, add five employees to your table, and work with the HIERARCHYID datatype to define their relationship to one another. The hierarchy manipulation methods you will discover follow:

- **GetRoot:** The GetRoot method assigns or returns the root of the hierarchy.
- **GetDescendant:** The GetDescendant method assigns or returns a child node in the hierarchy. It requires two nullable parameters: child1 and child2. If both child1 and child2 are Null, GetDescendant returns a child. If child1 is not Null, and child2 is Null, GetDescendant returns a child greater than child1.
- **GetAncestor:** The GetAncestor method returns a HierarchyID of the *n*th generation. For example, GetAncestor(1) will return child nodes, while GetAncestor(2) will return grandchildren.

These methods are found in the SqlHierarchyID class. You will use them to position and populate the employees table.

Creating A HIERARCHYID Table

First, create an employee containing a HIERARCHYID. The employee table will have the following attributes: a unique record identifier, the employee's name, the employee's position, and the employee's level in the company. Create the employee table in the AdventureWorksDW2008R2 database by entering and executing the following T-SQL code in an SSMS query:

```
Create Table dbo.EmployeeHierarchy
(
    EmployeeID Int Identity (1, 1) Not Null Primary Key,
    Name Varchar(50) Not Null,
    Position Varchar(50) Not Null,
    OrganizationLevel HierarchyID
);
Go
```

Adding Data to the Table

Next, you will add five employees to the `EmployeeHierarchy` table. The employees (and their positions within the company) will be named Jodi (CEO), Jim (CFO), Kay (COO), Bob (Manager), and Andy (Manager). To accomplish this, enter and execute the following code:

```
Declare @CEO HierarchyID,
        @COO HierarchyID,
        @CLevel HierarchyID;

Select @CEO = HierarchyID::GetRoot();

Insert Into dbo.EmployeeHierarchy (Name, Position, OrganizationLevel)
Values ('Jodi', 'CEO', @CEO),
       ('Jim', 'CFO', @CEO.GetDescendant(Null, Null));

Select @CLevel = MAX(OrganizationLevel)
From dbo.EmployeeHierarchy
Where OrganizationLevel.GetAncestor(1) = @CEO;

Insert Into dbo.EmployeeHierarchy (Name, Position, OrganizationLevel)
Values ('Kay', 'COO', @CEO.GetDescendant(@CLevel, Null));

Select @COO = OrganizationLevel
From dbo.EmployeeHierarchy
Where Position = 'COO';

Insert Into dbo.EmployeeHierarchy (Name, Position, OrganizationLevel)
Values ('Bob', 'Manager', @COO.GetDescendant(Null, Null)),
       ('Andy', 'Manager', @COO.GetDescendant(Null, Null));
```

The preceding example, which adds your five employees, begins by declaring three variables of type `HierarchyID`. Next, a `Select` statement uses `HierarchyID::GetRoot()` to create and assign a root node to `@CEO`. The following `Insert` statement adds Jodi as CEO with the `@CEO` `HierarchyID` as her `OrganizationLevel`. Inserting Jim into `EmployeeHierarchy` uses the `GetDescendant` method, with the `@CEO` `HierarchyID` as parent, making Jim report to Jodi.

Adding Kay as your COO is a two-step process. First, you use the `Select @CLevel` statement with `GetAncestor(1) = @CEO` to find the maximum `OrganizationLevel` below the CEO. Passing `@CLevel` to `GetDescendant` in Kay's `Insert` statement places Kay in your `EmployeeHierarchy` as a sibling of Jim. Kay and Jim now both report to Jodi.

Finally, you add Bob and Andy as managers. Both Bob and Andy report to Kay. By setting `@COO` to Kay's `OrganizationLevel` and using `@COO.GetDescendant`, you will add both Bob and Andy as direct reports (children) of Kay. Of course, Bob and Andy are now siblings. Figure A-1 displays an organization chart of the hierarchy you just created.

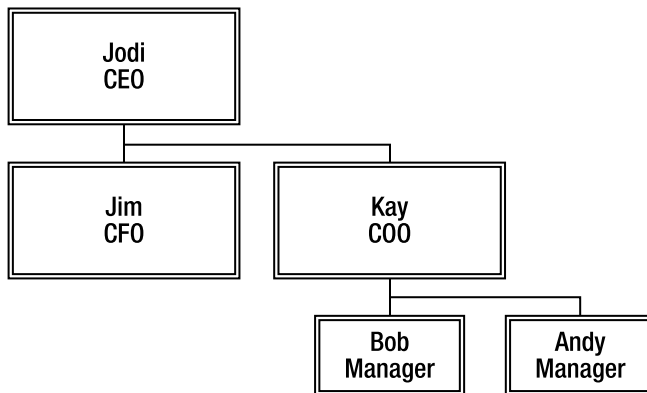


Figure A-1. The organization chart of employees you created in *EmployeeHierarchy*

Displaying Hierarchical Data in SSMS

You can view hierarchical data via SSMS. To view a textual representation of the *EmployeeHierarchy* table, enter and execute the following code:

```
Select Name, Position, OrganizationLevel.ToString()
From EmployeeHierarchy;
```

This will list each employee, along with his or her position and a text representation of that person's location in the hierarchy. The `ToString` method returns your `HierarchyID`, using a single slash character (/) to represent the root node. Looking at Bob and Andy, you can see that they are siblings and children of Kay. Figure A-2 shows the results of the preceding query in SSMS. The third column shows each person's location in the hierarchy. The delimited string that you see is termed a *materialized path*.

| | Name | Position | (No column name) |
|---|------|----------|------------------|
| 1 | Jodi | CEO | / |
| 2 | Jim | CFO | /1/ |
| 3 | Kay | COO | /2/ |
| 4 | Bob | Manager | /2/1/ |
| 5 | Andy | Manager | /2/1/ |

Query executed successfully. | GFAA (10.50 RTM) | GFAA\Guy (55) | AdventureWorksDW2008R2 | 00:00:00 | 5 rows

Ln 36 Col 1

Figure A-2. The organization chart of employees you created in *EmployeeHierarchy*, viewed in SSMS

Index



A

- account dimension, 50
- Accuracy and Validation group, in Microsoft Excel, 324
- Accuracy Chart button, 323
- Accuracy Chart wizard, 325
- actions, for OLAP modeling, 100–105
 - overview, 51
 - in SSAS, 100–104
 - in SSMS, 104
- Actions tab, BIDS, 100
- active record flag, 123
- Add Account Intelligence option, 126
- Add Attribute Ordering option, 129
- Add Business Intelligence Wizard, 94
- Add button, 296
- Add Calculated Field option, 296
- Add Copy of Existing Package dialog box, 173
- Add Currency Conversion option, 129
- Add Custom Member Formulas option, 129
- Add Dataset option, 285
- Add Dimension Intelligence option, 129
- Add Existing Package option, 173
- Add Model to Structure dialog box, 324
- Add Output dialog box, 317
- Add Semiadditive Measures option, 126
- Add Time Intelligence option, 127
- Add transformation group, 219
- Add Unary Operator option, 129
- Add Writeback option, 126
- Additional properties item, 358
- Advanced button
 - Define Relationship dialog box, 110
 - ribbon, 323
- Advanced Data Source View Options dialog box, 196
- Advanced page, 216
- Advanced Processing options page, 216
- Advanced property, 226, 233
- Advanced Query Editor panes, 317
- Advanced tab, Lookup Transformation Editor dialog box, 226
- advantages of MDX, 347
- Adventure Works
 - setting up BI, 10
 - using PowerPivot with, 336–346
 - adding calculated columns, 342–345
 - hiding columns, 340
 - importing data, 336–337
 - sorting and filtering, 338–340
 - using in Excel, 345–346
- Adventure Works cube, 16, 271–272
- aggregate dialog box, 219
- Aggregate function, 357, 366
- Aggregation Design Wizard, designing
 - aggregations using, 135–139
- Aggregation Storage setting, 152
- aggregationFunction property, 126
- aggregations
 - designing, 135–142

- using Aggregation Design Wizard, 135–139
 - using SQL Server Profiler, 141–142
 - using Usage-Based Optimization Wizard, 139–141
- with MOLAP, 133–134
- Aggregations tab, BIDS Cube Designer, 135
- algorithms
 - for data mining, 384–390
 - defined, 369
- Allow Duplicate Names property, 80
- Analysis Services Execute DDL task, 207
- Analysis Services Processing task, 207–208, 217
- Analysis Services Scripting Language (ASSL), 207
- architectural considerations, for data mining, 373–374
- Arrange Fields dialog box, 293
- ASSL (Analysis Services Scripting Language), 207
- Association category, 372
- Association Rules, Microsoft, 388
- Attribute Profiles view, 381
- Attribute Relationships tab, 76–78
- AttributeHierarchyOptimized property, 156
- AttributeHierarchyOptimizedState property, 156
- AttributeRelationship RelationshipType property, 156
- attributes, for dimension tables, 29–30
- Audit group, 219
- Audit task, 248
- Audit Transformation Editor dialog box, 227
- Automatic MOLAP option, 150
- Automatic option, 110
- AverageOfChildren aggregation function, 49

■ B

- Batch option, 103
- Begin Transaction statement, 264
- BI (Business Intelligence), 1–24
 - business problems addressed by, 22–23
 - concepts behind, 21–22
 - defined, 1–3
 - Microsoft tools for, 4–7
 - languages for, 7
 - overview, 5–7
 - reasons to use, 23
- setting up, 9–20
 - connecting to sample cube using Excel, 17–20
 - deploying standard edition of sample cube, 10–17
 - using AdventureWorks, 10
- BIDS (Business Intelligence Development Studio), 53–81
 - building cube, 67–70
 - defining cube, 61–64
 - defining hierarchies, 66–67
 - dimension attributes in
 - adding, 64–66
 - creating relationships for, 76–78
 - editing, 73–74
 - dimension properties, editing, 79–80
 - Dimension Structure tab in, 78
 - display format of measures, 70–72
 - offline vs. online mode in, 80–81
- BIDS browser interface exploring, 16–17
- BIDS Calculations tab, 364
- BIDS cube browser, 68
- BIDS window, 14
- BIGINT datatype, 153
- BirthDate column, 268
- BOL (Books Online), 132, 176
- Boolean datatype, 394
- break mode, 244–245
- Breakpoints option, BIDS, 245
- Browse button, 313, 316
- Browse window in Excel, 314
- Browser tab, Process Cube dialog, 69
- Build button, 37, 109, 113
- built-in Icon Set Conditional Formatting, 307
- Bulk Insert flow task, 204
- BULK INSERT statement, SQL, 204
- Business Intelligence. *See* BI
- Business Intelligence Development Studio. *See* BIDS
- Business Intelligence group, 219
- Business Intelligence Projects project type, 188

Business Intelligence Wizard, for OLAP modeling, 125–129
 business key, 27
 business problems, addressed by BI, 22–23
 ByAccount aggregation function, 49

■ C

CacheMode property, 374
 Calculated Columns SSIS sample package, 219
 calculated measures
 vs. derived measure, 50
 in MDX, 359–361
 Calculated Member Builder dialog box, 281, 283
 Calculated Members area, 281
 Calculated members feature, 355
 CalculatedColumns.dtsx package, 219, 221
 Calculation Tools section, Calculations tab, 358
 Calculations group, 333
 Calculations tab, 356
 Calendar Date hierarchy, 349
 Calendar Quarter Description level, 349
 Calendar Year attribute, 76–77
 Calendar Year cell, 285
 Calendar Year column, 273
 Calendar Year hierarchy, 16
 CalendarYear field, 64
 Candidate Key Profiles pane, 268
 CaptureDataLineage.dtsx package, 227
 case table, 390
 cFatResellerSales, 61
 Change filters pane, 294
 Change Settings button
 Dimension Key Errors tab, 112
 Process Cube dialog box, 158
 Changing Attribute, 46
 Changing Attribute option, 228
 CheckpointFileName property, 262
 checkpoints, in execution of SSIS packages, 261–262
 CheckpointUsage property, 262
 child1 parameter, 403
 child2 parameter, 403
 Choose a Data Source dialog box, 166
 Choose How to Import the Data dialog, 337
 Choose Perspective dialog box, 292
 Choose the Deployment Location dialog box, 284
 Choose the layout dialog box, 293
 Choose the Table Style dialog box, 284
 Choose Toolbox Items dialog box, 199
 City column, 75
 Classification category, 372
 Classification Matrix button, 323, 326
 Classification Matrix dialog box, 327
 Clean Data button, 319
 ClearAfterProcessing, 374
 Client Initiated option, Storage Options dialog box, 152
 Clipboard group, 333
 CLR (Common Language Runtime), 403
 Cluster Diagram view, 386
 CLUSTER_COUNT property, 387
 Clustering, Microsoft, 386
 CLUSTERING_METHOD, 386
 Color field, 65
 Column Chart viewer type, 241
 Column Length Distribution Profile, 266
 Column Null Ratio Profile, 265
 Column pane, 274
 Column Pattern Profile, 266
 Column Statistics Profile, 265
 Column Value Distribution Profile, 266
 Columns group, 333
 Commit Transaction statement, 264
 Common Language Runtime (CLR), 403
 completing the wizard wizard step, 64
 COMPLEXITY_PENALTY property, 377, 385
 components, of SSRS, 276–277
 concepts
 behind BI, 21–22
 of data mining, 372–373
 Conditional Formatting button, Home tab, 306
 Conditional Split dialog box, 221
 Conditional Split Transformation Editor dialog box, 223
 configurations, for remote partitions, 154
 Configure Dimension Attributes page, Business Intelligence Wizard, 126
 Configure SSIS Logs dialog box, 245

- Connection Manager dialog box, 265
- Connection Managers area
 - BIDS, 176
 - SSIS design surface, 195
- Connection Managers pane, 194
- Connection Managers window, 192
- Connection String list box, 278
- connections, for SSIS packages, 174–176, 193–196
- Connections group, 333
- Constraint option, 212
- container tasks, for SSIS packages, 200–201
- Containers group, 199
- content types, for data mining, 392–396
- contextual tab, Excel PivotTable Tools ribbon, 302
- contextual tabset, 302
- Continuous content type attribute, 393
- Control Flow area, 202, 214
- Control Flow design surface, 179, 197, 202
- Control Flow tab, 191
- Control Flow tasks, for SSIS packages
 - container tasks, 200–201
 - file system tasks, 204–205
 - operating system tasks, 205–206
 - overview, 176–178
 - precedence constraints for, 209–211
 - remote tasks, 207
 - script tasks, 206–207
 - SQL tasks, 201–204
 - SSAS tasks, 207–209
 - using expressions with, 211–214
- Control Flow toolbox, 200
- Control Flow work area, 238
- Copy data from one or more tables or views
 - option, Import and Export Wizard, 167
- Copy to Excel button, 315
- Count aggregation function, 49
- Count measures, 48
- CountRows function, 344
- Create Member, 361
- Create method, 131
- Create Mining Structure option, 324
- Create PivotTable dialog, 345
- Create Relationship button, 334

- Create Table dialog box, 180
- Create Trendline option, 308
- CreateDeploymentUtility property, 253
- CreateQueryLogTable property, 139
- Credentials button, 278
- CRUD (creating, reading, updating, and deleting), 403
- Cube term, 348
- cubes
 - building, using BIDS, 67–70
 - defining, using BIDS, 61–64
 - multiple fact tables in single, and OLAP modeling, 107–109
 - processing options for, 154–160
- Cubes folder, 15, 352
- Current function, 364
- current record flag, 123
- CurrentMember function, 87, 357
- Custom Options dialog box, 149
- Customer dimension, 32, 37, 75, 352
- CustomerAlternateKey identifier, 28
- CustomerCountByCountry.mht file, 275
- CustomerKey identifier, 28
- CustomRollupColumn property, 129
- Cyclical column, 393

■ D

- Data Bars button, 306
- Data Bars Conditional Formatting, 307
- Data Connection Wizard, 18–19, 272
- data definition language (DDL), 34
- Data Flow area, 203
- Data Flow design surface, 179, 241, 250
- Data Flow Script component, 245
- Data Flow tab, 191
- Data Flow tasks, for SSIS packages, 179–181, 214–219
 - adding transformations to, 182–186
 - data sources for, 214–215
 - destination types for, 216–217
 - transformation types for, 217–219
- Data Flow Transformation toolbox, 398
- Data Flow window, 202, 214
- Data Flow work area, 202, 214, 219, 238

- Data is not available to all users pain point, 23
- data mapping, for ETL design, 162–165
- data mart, 2
- Data member term, 348
- data mining, 369–402
 - algorithms for, 384–390
 - architectural considerations for, 373–374
 - concepts of, 372–373
 - content types for, 392–396
 - Data Mining Wizard, 390–392
 - datatypes for, 392–396
 - defined, 369–371
 - with Microsoft Excel, 308–328
 - Accuracy and Validation group in, 324
 - configuring Excel as client, 309–312
 - Data Mining tab in, 312–318
 - Data Modeling group in, 323–324
 - Data Preparation group in, 318–322
 - Mining Accuracy Chart tab, 381–382
 - Mining Model Prediction tab, 382–383
 - Mining Model Viewer tab, 378–381
 - Mining Models tab, 375–377
 - and OLAP, 50
 - processing mining models, 396–399
 - methods for, 396–397
 - and SSIS, 397–399
 - using DMX language, 399
- Data Mining Add-ins, 309, 323
- Data Mining Extensions (DMX), 7, 399
- Data Mining Query Task, 398
- Data Mining Query Transformation, 399
- Data Mining ribbon, 325
- Data Mining tab, in Microsoft Excel, 312–318
- Data mining using SSAS, 6
- Data Mining Wizard, 315, 390–392, 394–395
- Data Modeling group, in Microsoft Excel, 323–324
- Data Preparation group, in Microsoft Excel, 318–322
- Data Profile Viewer application, 268
- data profiling, in SSIS, 264–270
 - creating data profile, 264–267
 - viewing data profile, 267
- Data Profiling task, 267
- Data Profiling Task Editor, 264, 267
- Data Source Views (DSVs), 54, 115, 191, 196, 374, 396
- Data Source Views folder, 56, 173
- Data Source Wizard, 56, 174, 193
- data sources
 - for Data Flow tasks, 214–215
 - views as, with snowflake schema, 42
- Data Sources folder, Solution Explorer, 55, 173–174, 193, 195
- Data tab, Excel, 302
- data transformations, for SSIS packages, 219–234
 - enterprise edition-only transformations, 232–234
 - run command data transformations, 231–232
 - SCD, 227–231
 - split data transformations, 221–223
 - SSAS data transformations, 227
 - translate data transformations, 223–226
- data viewers, for SSIS packages, 241–243
- data warehouse, 2
- Database Engine Tuning Advisor (DETA), 248
- database owner (dbo), 152
- DataCleaning sample package, 237
- DataCleaning.dtsx package, 223, 232
- DataProfile.dtsx package, 264
- DataProfilerResults.xml file, 264, 267
- DataSet action type, 100
- Dataset node, 294
- Dataset Properties dialog box, 294
- Datasets node, Report Data pane, 285
- datatypes
 - for data mining, 392–396
 - HIERARCHYID datatype, 403–405
 - adding data to table, 404–405
 - creating table with, 403
 - displaying hierarchical data in SSMS, 405
- Date datatype, 394
- Date dimension, 76, 349
- Date function, 343
- Date Key attribute, 76
- dateLoaded column, 123

- DatesBetween function, 344
- Day level, 349
- Day Number Of Year attribute, 76–77
- db_dtsadmin role, 260
- db_dtsltduser role, 260
- db_dtsoperator role, 260
- dbo (database owner), 152
- DDL (data definition language), 34
- debugging, SSIS packages, 243–245
- Decision support system (DSS) database, 3
- Decision Tree view, 379
- Decision Trees, Microsoft, 385
- Default Member property, 80
- Default mining models process method, 396
- Default mining structures process method, 396
- DefaultMember property, 74
- Define Relationship dialog box, 40, 109–110, 113–115, 119
- degenerate dimensions, for OLAP modeling, 114
- Delivery Date dimension, 352
- denormalization, 4
- Dependency Network Viewer, 314
- Deploy SSIS Packages section, Package Installation Wizard, 254
- deployment options, for SSIS packages, 253–255
- DeploymentOutputPath property, 253
- Derived Column dialog box, 183
- Derived Column transformation, 183
- Derived Column Transformation Editor dialog box, 183
- derived measure, vs. calculated measure, 50
- Design Aggregations button, 135
- design mode, 280
- Design Mode button, 280
- Design tab, 59, 284–285, 333, 340
- Design the Query dialog box, 279
- Design the Table dialog box, 284
- designer, for PowerPivot for Excel, 332–335
- Destination drop-down list, 264
- destination types, for Data Flow tasks, 216–217
- DETA (Database Engine Tuning Advisor), 248
- Details tab, Configure SSIS Logs dialog box, 246
- Detect button, 392

- Deviation analysis category, 373
- Diff option, 204
- DiffGram format, 204
- DimCustomer dimension, 28
- DimDate dimension, 349
- DimEmployee table, 59, 180
- dimension attributes, in BIDS
 - adding, 64–66
 - creating relationships for, 76–78
 - editing, 73–74
- Dimension Key Errors tab, 112, 158
- Dimension Processing destination type, 216
- dimension properties, in BIDS, 79–80
- Dimension Structure tab, in BIDS, 64–67, 73, 78
- Dimension Structure work area, BIDS, 43
- dimension tables, in star schema, 27–30
 - attributes for, 29–30
 - keys for, 27–28
- Dimension term, 348
- Dimension Usage grid, 37
- Dimension Usage section, SSAS, 37
- Dimension Usage tab, cube designer work area, 108–109, 113, 119–120
- dimensional hierarchy, 44
- dimensions
 - for OLAP modeling, 112–122
 - degenerate dimensions, 114
 - dimensions that change, 123–124
 - error handling for, 124–125
 - many-to-many dimensions, 116–120
 - parent-child dimensions, 115–116
 - role-playing dimensions, 120–121
 - snowflake dimensions, 112–113
 - writeback dimensions, 121–122
 - processing options for, 154–160
- Dimensions folder, 64
- DimProductCategory, 334
- DimProductSubcategory, 334, 337
- Discrete content type attribute, 393
- DiscretizationBucketCount, 393
- DiscretizationMethod, 393
- Discretized content type attribute, 393
- Disparate data sources pain point, 23
- display format, of measures in BIDS, 70–72

- Display_Folder property, 90
- DistinctCount aggregation function, 49
- DistinctCount function, 364
- DistinctCount measures, 366
- Distributed Transaction Coordinator (DTC), 262
- DMX (Data Mining Extensions), 7, 399
- DMX templates, 401
- Do not design option, 138
- document type definition (DTD), 204
- DontSaveSensitive option, 260
- Double datatype, 394
- Drag Field area, 273
- drag-and-drop column fields area, 293
- Drillthrough Action, 102
- Drop KPI statement, 90
- Drop Member statement, 90
- DSS (Decision support system) database, 3
- DSVs (Data Source Views), 54, 115, 191, 196, 374, 396
- DTC (Distributed Transaction Coordinator), 262
- DTD (document type definition), 204
- dtexec.exe utility, 256, 260
- dtexecui.exe utility, 256
- dtutil.exe utility, 256
- Dynamic Package Configuration Wizard, using in SSIS, 234–235

■ E

- Edit Breakpoints button, 243
- Edit button, 278
- Edit Connection dialog, 333
- Edit Table Properties dialog, 335
- Employee attribute, 116
- Employee Department attribute, 16
- Employee dimension, 32
- Employee option, 59
- Employee table, 267, 270
- EmployeeHierarchy table, 405
- EmployeeKey attribute, 116
- EmployeeKey column, 184
- Enabled setting, 152
- EncryptAllWithKey option, 261
- EncryptAllWithPassword option, 261
- EncryptSensitiveWithPassword option, 261
- EncryptSensitiveWithUserKey option, 261
- English Product Category Name attribute, 66
- English Product Name attribute, 66
- English Product Subcategory Name attribute, 66
- EnglishProductCategoryName field, 65
- EnglishProductSubcategoryName field, 65
- Enterprise Edition group, 219
- Entities pane, 293
- EQUAL_AREAS, 393
- error handling
 - for dimensions, 124–125
 - for SSIS packages, 248–251
- Error List window, 289
- Error option, 111
- ErrorConfiguration property, 124
- Estimated storage reaches option, 137
- ETL (extract, transform, and load) design, using SSIS, 161–171, 187–188
 - data mapping for, 162–165
 - plan for, 161–162
 - and staging servers, 165
- Evaluation operation drop-down list, 212
- Event handler drop-down list, 251
- event handlers, for SSIS packages, 251–252
- Event Handlers tab, 191, 251
- Excel, Microsoft, 301–328
 - connecting to sample cube using, 17–20
 - data mining with, 308–328
 - Accuracy and Validation group in, 324
 - configuring Excel as client, 309–312
 - Data Mining tab in, 312–318
 - Data Modeling group in, 323–324
 - Data Preparation group in, 318–322
 - KPI support in, 306–308
 - PowerPivot for, 329–335
 - designer, 332–335
 - ribbon, 329–332
 - reporting with, 271–276
 - pivot charts, 274–275
 - pivot tables, 271–274
 - publishing workbook, 275–276
 - ribbon in, 301–304

- using PowerPivot data in, 345–346
- Excel Associate, 323
- Excel Classify, 323
- Excel Cluster, 323
- Excel Estimate, 323
- Excel Forecast, 323
- Excel PivotTable, 346
- Exchange Rate Measures column, 96
- Executable drop-down list, 251
- Execute DDL task, 208
- Execute Package option, 180
- Execute Package Utility dialog box, 171, 256, 259
- ExecuteSQLStatementInLoop.dtsx package, 221
- ExecuteSQLStatementsInLoop Sample folder, 197
- ExecuteSQLStatementsInLoop.dtsx file, 197, 221
- ExecuteSqlStatementsInLoop.sln file, 197
- execution, of SSIS packages, 237–252
 - options for, 256–260
 - using checkpoints in, 261–262
- Existing Connections icon, Get External Data group, 303
- Exists function, 367
- Explore Data button, 319–320
- Explore Data wizard, 319
- Export Data option, AdventureWorks2008R2 database, 166
- Expression and Constraint option, 212
- Expression area, Calculated Member Builder dialog box, 283
- Expression dialog box, 296
- Expression item, 358
- Expression option, 212
- Expression or Constraint option, 212
- Expression text box, 212
- expressions
 - in Control Flow tasks, for precedence constraints, 211–214
 - in SSIS, for properties, 236
- extract, transform, and load design, using SSIS. *See* ETL design, using SSIS

■ F

- fact (measure) modeling, 48–50
 - calculated measure vs. derived measure, 50
 - example of, 48–49
- fact dimension type, 122
- fact tables, multiple in single cube, 107–109
- FactInternetSales tab, 338
- FactInternetSales table, 338
- Fail Component option, 250
- FailPackageOnFailure property, 248, 262
- FailParentOnFailure property, 248
- False option, ForceRebuildInterval setting, 152
- Field Properties option, 294
- Fields box, 292
- Fields pane, 293
- File Connection Manager Editor, 264
- File drop-down list, 264
- file system tasks, for SSIS packages, 204–205
- Filter function, 365
- Filter tables text box, 143
- filtering, using PowerPivot with Adventure Works, 338–340
- Find Tables button, 143
- FirstChild aggregation function, 49
- FirstName column, 183
- FirstNonEmpty aggregation function, 49
- Fixed Attribute option, 228
- FK (foreign key), 26
- For loop, 200
- ForcedExecutionValue property, 248
- ForcedExecutionValueType property, 248
- ForceExecutionResult property, 249
- ForceExecutionValue property, 249
- ForceRebuildInterval setting, 152
- Foreach container task, 200
- Foreach loop, 200–201
- foreign key (FK), 26
- Foresach Loop Editor dialog box, 200
- Format Trendline dialog box, 308
- FormatString property, 70
- Freeze statement, 366–367
- FROM clause, 352
- full process option, 154
- Functional Dependency Profile, 266

functions, in MDX, 364–367
 Functions tab, 285
 Fuzzy Grouping transformation, 224
 Fuzzy Lookup transformation, 224
 fx button, 296

■ G

General system slowdowns pain point, 23
 Generate function, 365–366
 Generate Model button, 291
 Geography dimension, 37, 348
 GeographyKey attribute, Customer dimension, 37
 Get External Data group, 333
 GetAncestor method, 403
 GetDescendant method, 403
 GetRoot method, 403
 Getting Started dialog box, 325
 Getting Started wizard, 309
 Goal property, 85
 Google Reseller Search action, 104
 grain statements, creating star schema using, 33–34
 Greater Than Or Equal To menu item, 340
 Grid viewer type, 241
 Gross Profit Margin KPI, 90
 GrossMargin calculated column, 342

■ H

hard clustering, 386
 HIDDEN_NODE_RATIO parameter, 390
 Hide and Unhide Columns dialog, 340–341
 hiding columns, using PowerPivot with Adventure Works, 340
 hierarchies, defining in BIDS, 66–67
 Hierarchy term, 348
 HIERARCHYID datatype, 403–405
 adding data to table, 404–405
 creating table with, 403
 displaying hierarchical data in SSMS, 405
 HierarchyID::GetRoot() method, 404
 Histogram viewer type, 241
 Historical Attribute, 46
 Historical Attribute option, 228

Hit Count Type option, 244
 Hit Count Value option, 244
 HOLAP (Hybrid Online Analytical Processing), 143–146
 HOLAP cube type, 148
 HOLAP option, 146, 150
 Home tab, 333
 huge dimensions, storage for, 146–147
 Hybrid Online Analytical Processing (HOLAP), 143–146

■ I, J

I click stop option, 138
 Ignore Failure option, 250
 Ignore setting, 376
 IIF function, 364
 Immediate option, OnlineMode setting, 152
 Import and Export Wizard, 166, 173, 187, 193
 Import Data dialog box, 273
 Import Data option, AdventureWorks2008R2 database, 166
 Import Package button, 255
 importing data, using PowerPivot with Adventure Works, 336–337
 Input setting, 376
 Insert statement, 404
 Instructor dimension, 34
 InstructorDim table, 34
 int datatype, 26
 Integration Services option, Object Explorer, 170
 Integration Services Project project type, 172
 Interactive option, 103
 InternetSales dataset, 293, 296
 Invalid/inconsistent report data pain point, 23
 Invocation clause, 104
 Invocation drop-down list, 103
 IsolationLevel property, 262

■ K

KeepTrainingCases, 374
 Key column, 393
 Key Columns field, 268
 Key Performance Indicators. *See* KPIs
 Key Sequence column, 393

- Key Time column, 393
- Key Type column, 227
- Key Violations pane, 268
- KeyErrorAction property, 124
- keys, for dimension tables, 27–28
- KPICurrentTimeMember function, 367
- KPIGoal function, 85, 367
- KPIs (Key Performance Indicators)
 - OLAP, modeling, 83–90
 - in SSAS, 84–87
 - in SSMS, 89–90
 - overview, 51
 - support, in Microsoft Excel, 306–308
- KPIStatus function, 367
- KPITrend function, 367
- KPIValue function, 85, 367
- KPIWeight function, 367

■ L

- languages, for Microsoft tools for BI, 7
- language/translation view, 93
- Last change wins option, 46
- LastChild aggregation function, 49
- LastName column, 183
- LastNonEmpty aggregation function, 49
- Latency setting, 152
- Layout tab, 289
- lazyAggregations mode, 156
- Left function, 343
- Level term, 348
- lift chart, 381–382
- Linear Regression, Microsoft, 390
- linked dimension type, 122
- localizing measure values, OLAP modeling, 94–99
- Locals window, 244
- Log Events item, 191
- Log Events option, 246
- Log Events window, 191, 246
- Log providers configuration area, 258
- logging, execution results for SSIS packages, 245–248
- Logging option
 - BIDS, 245

- SSIS menu, 191
- LogicalAnd property, 210
- Logistic Regression, Microsoft, 390
- Long datatype, 394
- Lookup transformation, 223, 226
- Lookup Transformation Editor dialog box, 226
- Low-latency MOLAP option, 150

■ M

- Manage Models button, Management group, 313
- Manage Relationships button, 334
- Managing Mining Structures and Models dialog box, 313
- Manual query writing pain point, 23
- Many-to-Many dimensions, for OLAP modeling, 116–120
- Many-to-many option, 98
- Many-to-one option, 98
- Mappings page, 216
- market-basket analysis, 371–372
- Materialize check box, 40
- materialized path, 405
- Max aggregation function, 49
- MAXIMUM_INPUT_ATTRIBUTES parameter, 389
- MAXIMUM_ITEMSET_SIZE property, 388
- MAXIMUM_OUTPUT_ATTRIBUTES parameter, 389
- MDX (Multidimensional Expressions), 347–367
 - adding objects to cube, 356–358
 - advantages of, 347
 - calculated measures in, 359–361
 - common functions in, 364–367
 - named sets in, 361–362
 - query syntax for, 347–355
 - basic syntax, 350–353
 - and core terminology, 348–349
 - members, 353–355
 - sets, 353–355
 - tuples, 353–355
 - script commands for, 362–364
- MDX CHILDREN function, 354
- MDX scripts feature, 356
- MDX Template Explorer, 350

MDX work area, 283
 measure (fact) modeling, 48–50
 calculated measure vs. derived measure, 50
 example of, 48–49
 Measure Group Bindings dialog box, 110
 Measure Group Storage Settings dialog box, 150
 Measure Settings dialog, 331
 Measure term, 349
 measure values, localizing, 94–99
 MeasureExpression property, 71
 measures, display format of, 70–72
 Medium-latency MOLAP option, 150
 Member Names Unique property, 80
 members, in MDX, 353–355
 Members function, 362
 MemberValue function, 367
 Merge Join transformation, 219
 Merge option, 204
 Metadata area, Calculated Member Builder dialog box, 283
 Metadata browser, 280
 Metadata panel, 280
 Metadata tab, 285
 Metadata view, 285
 methods, for processing mining models, 396–397
 Microsoft Association Rules, 388
 Microsoft Clustering, 386
 Microsoft Decision Trees, 385
 Microsoft Excel. *See* Excel, Microsoft
 Microsoft Linear Regression, 390
 Microsoft Logistic Regression, 390
 Microsoft Message Queue (MSMQ), 206
 Microsoft Naïve Bayes, 384
 Microsoft Neural Network, 389
 Microsoft Sequence Clustering, 387
 Microsoft Time Series, 385
 Microsoft tools, for BI, 4–7
 languages for, 7
 overview, 5–7
 reasons to use, 23
 Mid function, 343
 Min aggregation function, 49
 Mining Accuracy Chart tab, 381–382

Mining Model Prediction tab, 382–383
 Mining Model Viewer tab, 378–381
 Mining Models tab, 375–377
 mining structures, 371
 Mining Viewer, 314
 Miscellaneous folder, 173
 MOLAP (default) option, 146
 MOLAP (Multidimensional Online Analytical Processing)
 aggregations with, 133–134
 vs. HOLAP, 143–146
 is default in SSAS, 135
 vs. ROLAP, 143–146
 XMLA for, 131–133
 MOLAP (nondefault) option, 146
 MOLAP cache, 149
 MOLAP cube type, 148
 MOLAP only option, Aggregation Storage setting, 152
 money datatype, 26
 Month level, 349
 Month Number Of Year attribute, 76–77
 MonthNumberOfYear field, 64
 More Rules button, 306
 MSDB (Multisource Database), 255
 MSMQ (Microsoft Message Queue), 206
 Multidimensional Expressions. *See* MDX
 Multidimensional Online Analytical Processing. *See* MOLAP
 multiple fact tables, in single cube, and OLAP modeling, 107–109
 Multisource Database (MSDB), 255

■ N

Naïve Bayes, Microsoft, 384
 Name property, 87
 Named set feature, 356
 named sets, in MDX, 361–362
 NameMatchingCriteria property, 61
 natural hierarchy, 75
 navigational hierarchy, 75
 nested table, 390
 Neural Network, Microsoft, 389
 New Calculated Member option, 281

- New Connection from Data Source option, 175, 194
- New Connection item, 192
- New Connection option, 194
- New Cube wizard, BIDS, 37, 44
- New Data Source option, Data Sources folder, 174
- New Partition link, 143
- New Project dialog box, BIDS, 54, 277
- New Report or Dataset dialog box, 293
- New Table or Matrix dialog box, 293
- NewID field, 34
- None aggregation function, 49–50
- nonstar dimensions, for OLAP modeling, 112–122
 - degenerate dimensions, 114
 - many-to-many dimensions, 116–120
 - parent-child dimensions, 115–116
 - role-playing dimensions, 120–121
 - snowflake dimensions, 112–113
 - writeback dimensions, 121–122
- Normalization, 4
- notifications, for proactive caching, 152–153
- Notifications tab, Storage Options dialog box, 152
- null values, and OLAP modeling, 109–112
- NullKeyConvertedToUnknown property, 111–112
- NullKeyNotAllowed property, 111–112
- Numeric_Expression, 317

■ O

- Object Explorer, 166, 352
- OfferingDim table, 34
- Office PivotTable interface, 59
- offline mode, vs. online mode, 80–81
- OLAP (Online Analytical Processing) modeling, 25, 51–53, 81–83, 105–129
 - actions for, 100–105
 - overview, 51
 - in SSAS, 100–104
 - in SSMS, 104
 - data mining, 50
 - dimensions that change, 123–124
 - error handling for dimensions, 124–125
 - fact (measure) modeling, 48–50
 - calculated measure vs. derived measure, 50
 - example of, 48–49
 - KPIs, 83–90
 - adding in SSAS, 84–87
 - adding in SSMS, 89–90
 - overview, 51
 - localizing measure values, 94–99
 - and multiple fact tables in single cube, 107–109
 - nonstar dimensions for, 112–122
 - degenerate dimensions, 114
 - many-to-many dimensions, 116–120
 - parent-child dimensions, 115–116
 - role-playing dimensions, 120–121
 - snowflake dimensions, 112–113
 - writeback dimensions, 121–122
 - and null values, 109–112
 - perspectives for, 51–91
 - snowflake schema, 37–42
 - overview, 37–41
 - using views as data sources with, 42
 - variations of, 42
 - when to use, 41–42
 - and source control, 51
 - star schema, 25–37
 - creating using grain statements, 33–34
 - dimension tables in, 27–30
 - overview, 26–27
 - tools for creating, 34–37
 - when to use, 30–33
 - and translations, 51
 - translations for, 92–94
 - UDM, 42–48
 - overview, 43–44
 - RCD, 47–48
 - SCD, 45–46
 - writeback with, 48
 - using BIDS, 53–81
 - adding dimension attributes, 64–66
 - building cube, 67–70
 - creating attribute relationships, 76–78
 - defining cube, 61–64

- defining hierarchies, 66–67
 - Dimension Structure tab in, 78
 - display format of measures, 70–72
 - editing dimension attributes, 73–74
 - editing dimension properties, 79–80
 - offline vs. online mode in, 80–81
 - using Business Intelligence Wizard for, 125–129
- OldID field, 34
- OLTP (online transaction processing)
 - databases, 3
- On Columns clause, 285, 352
- On Open option, 103
- on Rows clause, 352
- OnCacheComplete option, OnlineMode
 - setting, 152
- OnError event handler, 251
- One-to-many option, 98
- OnExecStatusChanged event handler, 251
- OnInformation event handler, 251
- Online Analytical Processing modeling. *See* OLAP modeling
- online mode, vs. offline mode, 80–81
- online transaction processing (OLTP)
 - databases, 3
- OnlineMode setting, 152
- OnPostExecute event handler, 251
- OnPostValidate event handler, 251
- OnPreExecute event handler, 252
- OnPreValidate event handler, 252
- OnProgress event handler, 251–252
- OnQueryCancel event handler, 252
- OnTaskFailed event handler, 246, 251–252
- OnVariableValueChanged event handler, 252
- OnWarning event handler, 252
- OPENQUERY operator, 399
- operating system tasks, for SSIS packages, 205–206
- Options button, 151
- Order By property, 80
- Order Count measure, 16
- OrderDateKey column, 338–339
- Ordered column, 393
- OrderQuantity column, 26
- outlier cases, 373

Outliers dialog box, 319

■ P

- Package Configuration Wizard, 191, 234
- Package Configurations option
 - BIDS, 234
 - SSIS menu, 191
- Package Explorer view, 237
- Package Installation Wizard, 254
- Package location drop-down list, Add Copy of
 - Existing Package dialog box, 173
- package Progress window view, 237
- Package Validation section, Package
 - Installation Wizard, 254
- PackagePassword property, 261
- packages, for SSIS, 171–181, 188–196
 - adding Control Flow tasks to, 176–178
 - adding transformations to, 182–186
 - configuring connections for, 193–196
 - configuring Data Flow tasks for, 179–181
 - connections for, 174–176
 - creating new, 188–193
 - data viewers for, 241–243
 - debugging SSIS packages, 243–245
 - deployment options for, 253–255
 - DSVs for, 196
 - error handling for, 248–251
 - event handlers for, 251–252
 - execution options for, 256–260
 - logging execution results, 245–248
 - overview of execution, 237–252
 - project for, 172–173
 - security options for, 260–261
 - using checkpoints in execution of, 261–262
 - using transactions in, 262–264
- pain points, 22
- ParallelPeriod function, 87, 347, 366
- Parameter Properties option, 286
- Parameters node, Report Data pane, 286
- Parent properties item, 358
- parent-child dimensions, for OLAP modeling, 115–116
- parse button, 284
- partition function, 154

- Partition Processing data flow destination type, 217
- partitioning storage, of cubes
 - relational table partitioning, 153–154
 - remote partition configurations, 154
- Partitions tab, 145
- Patch option, 204
- Performance gain reaches option, 137–138
- PERIODICITY_HINT property, 385
- PeriodsToDate function, 357
- PeriodToDate function, 357
- [Person].[AddressType] table, 168
- perspectives
 - and OLAP, 51
 - for OLAP modeling, 91
- Pivot Chart button, ribbon, 308
- pivot charts, reporting with Excel, 274–275
- pivot tables, reporting with Excel, 271–274
- PivotTable button, Reports group, 345
- PivotTable Field List, 273, 304–305
- PivotTable Report data view, 273
- PivotTable Tools menu area, 274
- PK (primary key), 27
- PowerPivot, 329–346
 - for Excel, 329–335
 - designer, 332–335
 - ribbon, 329–332
 - using with Adventure Works, 336–346
 - adding calculated columns, 342–345
 - hiding columns, 340
 - importing data, 336–337
 - sorting and filtering, 338–340
 - using in Excel, 345–346
- PowerPivot Designer, 330, 338
- PowerPivot Field List, 345
- PowerPivot tab, 330
- PowerPivot Window, 336
- Precedence Constraint Editor dialog box, 211
- precedence constraints, in Control Flow tasks
 - expressions in, 211–214
 - overview, 209–211
- Predict function, 317, 383
- Predict setting, 376
- PredictAssociation, 383
- PredictOnly setting, 376
- PredictProbability function, 317, 383
- PredictSupport function, 317
- Prepare Query button, 283–284
- Preserve option, 110
- Preview design surface, 289
- Preview tab, 284, 289
- PrevMember function, 90
- primary key (PK), 27
- proactive caching, 145, 149–153
 - notifications for, 152–153
 - settings for, 151–152
- Process Clear Structure mining structures
 - process method, 396
- Process command, 397
- Process Cube dialog box, 67, 69, 158–159
- Process Data option, 156
- Process Database dialog box, 14
- Process Default option, 156
- Process Dimension (or cube) dialog box, BIDS, 112
- Process Full mining models process method, 396
- Process Full mining structures process method, 396
- Process Full option, 156
- Process Incremental (cubes only) option, 156
- Process Index option, 156
- Process Progress dialog box, BIDS, 69
- Process Structure (cubes only) option, 156
- Process Structure mining structures process method, 396
- Process Unprocess mining models process method, 396
- Process Update (dimensions only) option, 156
- processing mining models, 396–399
 - methods for, 396–397
 - and SSIS, 397–399

Processing Options tab, Process Cube dialog box, 158
 Product dimension, 37, 67
 Product Lines, 362
 Profiles (Table View) pane, 268
 profit chart, 382
 Profit Chart button, 327
 Progress pane, 184
 Progress tab, 192, 239
 Progress window, 239
 projects, for SSIS packages, 172–173
 properties, expressions in SSIS for, 236
 Properties dialog box, 156, 248
 Properties window, 72, 79
 property page, XML source editor, 215
 Property Pages dialog box, 287
 Proprietary action type, 100
 ProtectionLevel options, 261
 ProtectionLevel property, 260

■ Q

Query Builder button, 279
 Query Builder interface, 283
 Query button, 315
 Query Designer interface, 279–280, 285
 Query Model wizard, 315–316
 Query Optimization Wizard, 158
 Query Parameters button, 283
 Query Parameters dialog box, 283
 query syntax, for MDX, 347–355
 basic syntax, 350–353
 and core terminology, 348–349
 members, 353–355
 sets, 353–355
 tuples, 353–355
 QueryLogConnectionString property, 139
 QueryLogSampling property, 139
 Quick Profile button, 265

■ R

rapidly changing dimensions (RCDs), for UDM, 47–48
 Raw File Source type, 215

RCDs (rapidly changing dimensions), for UDM, 47–48
 RDBMS (relational database management system), 26
 RDL (Report Definition Language), 288
 RDMS (relational database management system), 262
 Redirect Row option, 250
 referenced dimension type, 122
 Regression or forecasting category, 373
 Regular option, Aggregation Storage setting, 152
 Related function, 343
 relational database management system (RDBMS), 26
 relational database management system (RDMS), 262
 Relational Online Analytical Processing. *See* ROLAP
 relational table partitioning, 153–154
 Relationship section, Measure Group Bindings dialog box, 110
 relationships, creating for dimension attributes, 76–78
 remote partitions, configurations for, 154
 remote tasks, for SSIS packages, 207
 Report Builder button, Report Manager toolbar, 292
 Report Builder, reporting with, 290–299
 creating dataset, 292–293
 creating report, 293–299
 creating report model, 290–292
 Report Builder tool, 298–299
 Report Data pane, 285–286, 294
 Report Definition Language (RDL), 288
 Report Designer, 277
 Report group, 331
 Report Manager tool, 276, 298
 Report Manager toolbar, 292
 Report Manager Web interface, 290
 Report Manager Web site, 290
 Report Model Project template, 277
 Report Parameters Properties dialog box, 286
 Report Properties dialog box, 297
 Report Server Project template, 277

- Report Server Project Wizard, 278–290
 - designing query, 279–284
 - previewing and designing report, 284–287
 - publishing report, 287–290
- Report Server Project Wizard template, 277
- Report Server Web Service, 277, 287
- reporting tools, 271–299
 - Excel, 271–276
 - pivot charts, 274–275
 - pivot tables, 271–274
 - publishing workbook, 275–276
 - Report Builder, 290–299
 - creating dataset, 292–293
 - creating report, 293–299
 - creating report model, 290–292
 - SSRS, 276–290
 - components of, 276–277
 - samples for, 277
 - using Report Server Project Wizard, 278–290
- Reports group, 333
- Retain all history option, 46
- Retain some history option, 46
- RetainSameConnection property, 264
- Review Aggregation Usage dialog box, 136–137
- Revision Number - Fact Reseller Sales, 61
- ribbon
 - in Microsoft Excel, 301–304
 - for PowerPivot for Excel, 329–332
- Right function, 343
- ROLAP (Relational Online Analytical Processing)
 - vs. HOLAP, 143–146
 - vs. MOLAP, 143–146
 - overview, 142–143
- ROLAP cube type, 148
- ROLAP dimension type, 148
- ROLAP mode, 151
- ROLAP option, 146, 151
- role-playing dimensions, for OLAP modeling, 120–121
- Rollback Transaction statement, 264
- Row groups area, 219, 293
- rowguid column, 169
- Rowset action type, 100

- Rowset group, 219
- Run button, 298
- run command data transformations, for SSIS packages, 231–232
- Run immediately check box, 169
- Run Package option, 171, 256

■ S

- Sales Amount measure, 69
- Sales Quota Allocation script, 362
- SalesOrderNumber column, 26
- sample cube
 - connecting to using Excel, 17–20
 - deploying standard edition of, 10–17
- Sample Data button, 321
- Sample Data wizard, 321–322
- Save As dialog box, 275
- Save as type drop-down list, 275
- Save SSIS Package dialog box, 169
- SaveCheckpoints property, 262
- Scatter Plot viewer type, 241
- SCD (Slowly Changing Dimension)
 - for SSIS packages, 227–231
 - for UDM, 45–46
- Scheduled MOLAP option, 150
- Scheduled Polling option, Storage Options dialog box, 153
- Scheduling and Delivery Processor, 277
- Scope keyword, 362
- Scope statement, 362, 366
- script commands, for MDX, 362–364
- Script Design window, 245
- Script group, 199
- Script Organizer, 356
- Script task, 232
- script tasks, for SSIS packages, 206–207
- Script View button, BIDS, 360
- security options, for SSIS packages, 260–261
- Segmentation or clustering category, 372
- Select Certificate dialog box, 261
- SELECT clause, 352
- select creation method wizard step, 64
- SELECT DISTINCT statement, 44
- SELECT FROM <model> (DML) statement, 383

- Select how to define the connection dialog box, 174, 193
- select measure group tables wizard step, 64
- select measures wizard step, 64
- Select Members page, Add Business Intelligence Wizard, 96
- select new dimensions wizard step, 64
- Select Relationship Type drop-down list, 39, 113
- Select Source Tables and Views dialog box, 168
- Select statement, 404
- Select Tables and Views dialog, 58, 337
- Select the Report Type dialog box, 284
- Semantic Model Definition Language (SMDL), 290
- Send Mail Task Editor configuration dialog box, 178
- Sequence analysis and prediction category, 373
- Sequence Clustering, Microsoft, 387
- Sequence tasks, 200
- ServerStorage option, 261
- Set Aggregation Options dialog box, 137–138
- Set All to Default button, 136
- Set Currency Conversion options page, Add Business Intelligence Wizard, 95
- "Set Default Member - Attribute Name" dialog box, 74
- Set Default Member dialog box, 74
- sets, in MDX, 353–355
- settings, for proactive caching, 151–152
- SetToArray function, 366
- SharePoint, 6
- SharePoint Portal Server (SPS), 81
- Show subtotals and grand totals check box, 293
- Sign button, 261
- silence interval property, 150
- silence override interval property, 150
- SilenceInterval setting, 152
- SilenceOverrideInterval setting, 152
- Simple Object Access Protocol (SOAP), 131
- single cube, multiple fact tables in, 107–109
- Single File Web Page option, Save as type drop-down list, 275
- Single Table Quick Profile Form, 265–266
- Slowly Changing Dimension. *See* SCD
- Slowly Changing Dimension Wizard, 227
- Slow-to-execute queries pain point, 23
- SMDL (Semantic Model Definition Language), 290
- SMEs (subject matter experts), 33, 163
- SMO (SQL Management Objects), 201
- snowflake dimensions, for OLAP modeling, 112–113
- snowflake schema modeling, 37–42
 - overview, 37–41
 - using views as data sources with, 42
 - variations of, 42
 - when to use, 41–42
- SOAP (Simple Object Access Protocol), 131
- soft clustering, 386
- Solution Explorer window, 287
- SOLVE_ORDER keyword, 363
- Sort and Filter group, 333
- Sort transformation, 183
- sorting, using PowerPivot with Adventure Works, 338–340
- Sorting dialog box, 295
- source control, and OLAP, 51
- Specify Object Counts dialog box, 137
- Specify Profit Chart Parameters dialog box, 327
- split data transformations, for SSIS packages, 221–223
- Split/Join group, 219
- SPS (SharePoint Portal Server), 81
- SQL Management Objects (SMO), 201
- SQL Server 2008 R2, 5
- SQL Server Analysis Services. *See* SSAS
- SQL Server group, 199
- SQL Server Integration Services. *See* SSIS
- SQL Server Management Studio. *See* SSMS
- SQL Server option, Storage Options dialog box, 152
- SQL Server Profiler, designing aggregations using, 141–142
- SQL Server Reporting Services. *See* SSRS
- SQL tasks, for SSIS packages, 201–204
- SqlHierarchyID class, 403
- SSAS (SQL Server Analysis Services)
 - actions for OLAP modeling in, 100–104
 - adding KPIs in, 84–87

- SSAS connection, 291
- SSAS cubes, 306
- SSAS data sources, 316, 322
- SSAS data transformations, for SSIS packages, 227
- SSAS database, 304
- SSAS Dimension Usage tab, BIDS cube designer, 32
- SSAS group, 199
- SSAS metadata browser, 351
- SSAS project, 322
- SSAS queries, 322
- SSAS source, 302
- SSAS tasks, for SSIS packages, 207–209
- SSIS (SQL Server Integration Services), 161, 186–187, 236–237, 270
 - Control Flow tasks, 198–214
 - container tasks, 200–201
 - file system tasks, 204–205
 - operating system tasks, 205–206
 - precedence constraints for, 209–211
 - remote tasks, 207
 - script tasks, 206–207
 - SQL tasks, 201–204
 - SSAS tasks, 207–209
 - using expressions with, 211–214
 - Data Flow tasks, 214–219
 - data sources for, 214–215
 - destination types for, 216–217
 - transformation types for, 217–219
 - data profiling in, 264–270
 - creating data profile, 264–267
 - viewing data profile, 267
 - data transformations, 219–234
 - enterprise edition-only transformations, 232–234
 - run command data transformations, 231–232
 - SCD, 227–231
 - split data transformations, 221–223
 - SSAS data transformations, 227
 - translate data transformations, 223–226
- ETL design, 161–171, 187–188
 - data mapping for, 162–165
 - plan for, 161–162
 - and staging servers, 165
- expressions in, for properties, 236
- packages for, 171–181, 188–196
 - adding Control Flow tasks to, 176–178
 - adding transformations to, 182–186
 - configuring connections for, 193–196
 - configuring Data Flow tasks for, 179–181
 - connections for, 174–176
 - creating new, 188–193
 - data viewers for, 241–243
 - debugging SSIS packages, 243–245
 - deployment options for, 253–255
 - DSVs (data source views) for, 196
 - error handling for, 248–251
 - event handlers for, 251–252
 - execution options for, 256–260
 - logging execution results, 245–248
 - overview of execution, 237–252
 - project for, 172–173
 - security options for, 260–261
 - using checkpoints in execution of, 261–262
 - using transactions in, 262–264
- processing mining models with, 397–399
- sample packages included, 197
- using Dynamic Package Configuration Wizard in, 234–235
- SSIS Import and Export Wizard option, BIDS, 166
- SSIS menu, 191
- SSIS Package Installation Wizard, 253
- SSIS Packages folder, Solution Explorer, 173
- SSIS Slowly Changing Dimension wizard, 46
- SSMS (SQL Server Management Studio)
 - actions for OLAP modeling in, 104
 - adding KPIs in, 89–90
 - displaying hierarchical data in, 405
- SSMS T-SQL query mode, 284
- SSRS (SQL Server Reporting Services), 276–290
 - components of, 276–277
 - samples for, 277
 - using Report Server Project Wizard, 278–290
 - designing query, 279–284
 - previewing and designing report, 284–287

- publishing report, 287–290
- Stacked Column in 3D option, Column pane, 274
- staging servers, and ETL design, 165
- star schema modeling, 25–37
 - creating using grain statements, 33–34
 - dimension tables in, 27–30
 - attributes for, 29–30
 - keys for, 27–28
 - overview, 26–27
 - tools for creating, 34–37
 - when to use, 30–33
- State attribute, 75
- State column, 75
- State Transitions view, 387
- Statement action type, 100
- Status expression, 364
- Stop Debugging button, 238
- storage for cubes, 131–160
 - designing aggregations for, 135–142
 - using Aggregation Design Wizard, 135–139
 - using SQL Server Profiler, 141–142
 - using Usage-Based Optimization Wizard, 139–141
- HOLAP
 - vs. MOLAP, 143–146
 - overview, 143
 - vs. ROLAP, 143–146
- for huge dimensions, 146–147
- MOLAP, 131–135
 - aggregations with, 133–134
 - vs. HOLAP, 143–146
 - is default in SSAS, 135
 - vs. ROLAP, 143–146
 - XMLA for, 131–133
- partitioning for, 153–154
 - relational table partitioning, 153–154
 - remote partition configurations, 154
- and proactive caching, 149–153
 - notifications for, 152–153
 - settings for, 151–152
- ROLAP
 - vs. HOLAP, 143–146
 - vs. MOLAP, 143–146

- overview, 142–143
- summary of, 148
- Storage Options dialog box, 152
- Storage Settings dialog box, 145
- Storage Settings link, Partitions tab, 145
- StorageMode property, 123, 146
- Student dimension, 34
- StudentDim table, 34
- Styles pane, 293
- subject matter experts (SMEs), 33, 163
- Sum aggregation function, 48–49
- surrogate key, 28
- SurveyDim table, 34
- Sync Partitions package, 208
- Sync Partitions.dtsx package, 206
- SyncAdvWorksPartitions solution, 206, 208

■ T

- Table Import Wizard dialog, 336
- Table or Matrix Wizard, 293
- Table or View drop-down list, 265
- Table Properties dialog, 335
- Tablix Properties option, 295
- Template Explorer, 400
- Templates tab, 285
- Term Extraction transformation, 233
- Test button, 212
- Test Connection button, 13, 337
- TestVar variable, 190
- TexasManagers group, 80
- Text datatype, 394
- Time Series, Microsoft, 385
- Too much data pain point, 23
- ToString method, 405
- TransactionOption property, 262, 264
- transactions, in SSIS packages, 262–264
- transformation types, for Data Flow tasks, 217–219
- transformations
 - for Data Flow tasks, 182–186
 - for SSIS packages, 219–234
 - enterprise edition-only transformations, 232–234

- run command data transformations, 231–232
- SCD, 227–231
- split data transformations, 221–223
- SSAS data transformations, 227
- translate data transformations, 223–226
- translate data transformations, for SSIS packages, 223–226
- translations, for OLAP modeling, 51, 92–94
- True option, ForceRebuildInterval setting, 152
- tuples, in MDX, 353–355
- Type 1 standard solution, 46
- Type 2 standard solution, 46
- Type 3 standard solution, 46
- type property, 129

■ U

- UDM (Unified Dimension Modeling), 42–48
 - overview, 43–44
 - RCD, 47–48
 - SCD, 45–46
 - writeback with, 48
- Union transformation, 223
- Unknown Member option, 111
- UnknownMember property, 79, 124
- UnknownMember value, 367
- Unorder function, 367
- Unprocess mining structures process method, 396
- Unprocess option, 156
- URL action type, 100
- Usage Based Optimization button, 139–140
- Usage type drop-down list, 264
- Usage-Based Optimization Wizard, designing aggregations using, 139–141
- Use Windows Authentication (Integrated Security) connection type, 278

■ V

- Validate option, 204
- Value property, 85, 87
- Values area, 293
- Variables item, 191
- very large databases (VLDBs), 153

- View Code option, 252, 288
- View group, 333
- View item, 192
- views, as data sources, 42
- Visual Source Safe (VSS), 51, 81
- Visual Studio 2008 Tools for Applications (VSTA), 206
- Visual Studio Team System (VSTS), 81
- Visual Studio (VS), 171
- VLDBs (very large databases), 153
- VS (Visual Studio), 171
- VSS (Visual Source Safe), 51, 81
- VSTA (Visual Studio 2008 Tools for Applications), 206
- VSTS (Visual Studio Team System), 81
- vTargetMail view, 374

■ W

- Watch window, 244
- Welcome dialog box, 166, 174
- WHERE clause, 352
- Windows Management Instrumentation (WMI), 205
- With Member, 361
- WMI (Windows Management Instrumentation), 205
- WMI Data Reader task, 205
- WMI Event Watcher task, 205
- WMI Query Language (WQL), 205
- Work Offline item, 191
- Workflow Tasks group, 199
- WQL (WMI Query Language), 205
- Write a query to specify the data to transfer option, Import and Export Wizard, 168
- writeback, with UDM, 48
- writeback dimensions, for OLAP modeling, 121–122
- Writeback property, 362
- WriteEnabled property, 121, 126

■ X

- XML for Analysis (XMLA), for MOLAP, 131–133
- XML schema document (XSD), 204
- XML Schema document (XSD), 288

XMLA (XML for Analysis), for MOLAP
 (Regular) Action, 80, 100
 overview, 131–133
XPath operation type, 204
XPath option, 204
XSD (XML schema document), 204

XSD (XML Schema document), 288
XSLT option, 204
xxxKey column, 26

■ **Y, Z**

Yearly Income column, 375

Foundations of SQL Server 2008 R2 Business Intelligence

Second Edition



Guy Fouché
Lynn Langit

Apress®

Foundations of SQL Server 2008 R2 Business Intelligence, Second Edition

Copyright © 2011 by Guy Fouché and Lynn Langit

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13 (pbk): 978-1-4302-3324-4

ISBN-13 (electronic): 978-1-4302-3325-1

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

President and Publisher: Paul Manning

Lead Editor: Jonathan Gennick

Technical Reviewer: Michael Coles

Editorial Board: Steve Anglin, Mark Beckner, Ewan Buckingham, Gary Cornell, Jonathan Gennick, Jonathan Hassell, Michelle Lowman, Matthew Moodie, Jeff Olson, Jeffrey Pepper, Frank Pohlmann, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh

Coordinating Editor: Anita Castro

Copy Editor: Heather Lang and Tiffany Taylor

Compositor: MacPS, LLC

Indexer: BIM Indexing & Proofreading Services

Artist: April Milne

Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media, LLC., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales—eBook Licensing web page at www.apress.com/info/bulksales.

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

To Jodi Fouché: for her poetry, being my biggest fan, and unequivocal love

—Guy Fouché

Contents

| | |
|--|-------------|
| Contents at a Glance | iv |
| About the Authors | xiii |
| About the Technical Reviewer | xiv |
| Acknowledgments | xv |
| | |
| ■ Chapter 1: What Is Business Intelligence? | 1 |
| Just What Is Business Intelligence? | 1 |
| Defining BI Using Microsoft's Tools | 4 |
| What Microsoft Products Are Involved? | 5 |
| BI Languages | 7 |
| Understanding BI from an End User's Perspective | 9 |
| Building the First Sample—Using AdventureWorks | 10 |
| Deploying the Standard Edition Version of the Sample Cube | 10 |
| To deploy the Standard edition of the sample cube, follow these steps: | 10 |
| How to Connect to the Sample Cube Using Excel | 17 |
| Understanding BI Through the Sample | 21 |
| Understanding the Business Problems That BI Addresses | 22 |
| Reasons to Switch to Microsoft's BI Tools | 23 |
| Summary | 24 |
| | |
| ■ Chapter 2: OLAP Modeling Concepts | 25 |
| Modeling OLAP Source Schemas—Stars | 25 |
| Understanding the Star Schema | 26 |
| Understanding a Dimension Table | 27 |
| Why Create Star Schemas? | 30 |
| Effectively Creating Star Schema Models Using Grain Statements | 33 |

| | |
|---|-----------|
| Tools for Creating Your OLAP Model | 34 |
| Modeling Source Schemas—Snowflakes and Other Variations | 37 |
| Understanding the Snowflake Schema | 37 |
| Knowing When to Use Snowflakes | 41 |
| Considering Other Possible Variations | 42 |
| Choosing Whether to Use Views Against the Relational Data Sources | 42 |
| Understanding Unified Dimensional Modeling | 42 |
| Using the UDM | 43 |
| The Slowly Changing Dimension (SCD) | 45 |
| Type 1, 2, 3 SCD Solutions | 46 |
| The Rapidly Changing Dimension (RCD) | 47 |
| Writeback Dimension | 48 |
| Understanding Fact (Measure) Modeling | 48 |
| An Example | 48 |
| Calculated Measure vs. Derived Measure | 50 |
| Other Types of Modeling | 50 |
| Data Mining | 50 |
| Key Performance Indicators | 51 |
| Actions, Perspectives, Translations | 51 |
| Source Control and Other Documentation Standards | 51 |
| Summary | 51 |
| ■ Chapter 3: Introducing OLAP Modeling with SSAS | 53 |
| Using BIDS to Build a Cube | 53 |
| Defining Your First Cube | 61 |
| Adding Dimension Attributes | 64 |
| Defining Hierarchies | 66 |
| Building Your First Cube | 67 |
| Refining Your Cube | 70 |
| Reviewing Measures | 70 |
| Reviewing Dimensions: Attributes | 73 |
| Reviewing Dimensions: Hierarchies | 75 |
| Summary | 81 |

| | |
|---|------------|
| Chapter 4: Intermediate OLAP Modeling with SSAS..... | 83 |
| Adding Key Performance Indicators (KPIs)..... | 83 |
| Implementing KPIs in SSAS | 84 |
| Implementing KPIs in SSMS..... | 89 |
| Using Perspectives and Translations | 91 |
| Perspectives..... | 91 |
| Translations | 92 |
| Localizing Measure Values..... | 94 |
| Using Actions | 100 |
| Creating Actions in SSAS | 100 |
| Creating Actions in SSMS | 104 |
| Summary..... | 105 |
| Chapter 5: Advanced OLAP Modeling with SSAS | 107 |
| Multiple Fact Tables in a Single Cube..... | 107 |
| Nulls | 109 |
| Nonstar Dimensions..... | 112 |
| Snowflake Dimensions..... | 112 |
| Degenerate Dimensions | 114 |
| Parent-Child Dimensions | 115 |
| Many-to-Many Dimensions..... | 116 |
| Role-Playing Dimensions | 120 |
| Writeback Dimensions | 121 |
| Dimensions That Change | 123 |
| Error Handling for Dimension Attribute Loads | 124 |
| Using the Business Intelligence Wizard | 125 |
| Summary..... | 129 |
| Chapter 6: Cube Storage and Aggregation | 131 |
| Using the Default Storage: MOLAP..... | 131 |
| XML for Analysis | 131 |
| Aggregations | 133 |

| | |
|---|----------------|
| MOLAP as Default in SSAS | 135 |
| Adding Aggregations | 135 |
| The Aggregation Design Wizard | 135 |
| The Usage-Based Optimization Wizard | 139 |
| The SQL Server Profiler as an Aggregation Design Aid | 141 |
| Using Advanced Storage | 142 |
| Understanding ROLAP | 142 |
| Understanding HOLAP | 143 |
| Considering Non-MOLAP Storage | 143 |
| Handling Huge Dimensions | 146 |
| Summarizing OLAP Storage Options | 148 |
| Using Proactive Caching | 149 |
| Fine-Tuning Proactive Caching | 151 |
| Setting Notifications for Proactive Caching | 152 |
| Deciding Between OLTP and OLAP Partitioning | 153 |
| Relational Table Partitioning in SQL Server | 153 |
| OLAP Partition Configurations | 154 |
| Choosing Cube and Dimension Processing Options | 154 |
| Summary | 159 |
| ■ Chapter 7: Introducing SSIS | 161 |
| Understanding ETL | 161 |
| Creating a Plan | 161 |
| Creating a Data Map | 162 |
| Refining a Data Map | 164 |
| Adding a Staging Server | 165 |
| Creating a Basic SSIS Package | 166 |
| Building Basic SSIS Packages | 171 |
| Developing SSIS Packages | 172 |
| Designing SSIS Packages | 174 |
| Adding Transformations to the Data Flow | 182 |
| Summary | 186 |

| | |
|--|------------|
| ■ Chapter 8: Intermediate SSIS | 187 |
| Common ETL Package-Design Practices | 187 |
| Creating an SSIS Package from Scratch | 188 |
| Creating the Package Itself | 188 |
| Configuring Connections | 193 |
| Using Data Source Views (DSVs) | 196 |
| Reviewing the Included Samples Packages | 197 |
| Adding Control Flow Tasks | 198 |
| Container Tasks | 200 |
| SQL Tasks | 201 |
| File System Tasks | 204 |
| Operating System Tasks | 205 |
| Script Tasks | 206 |
| Remote Tasks | 207 |
| SSAS Tasks | 207 |
| Precedence Constraints | 209 |
| Using Expressions with Precedence Constraints | 211 |
| Understanding Data Flow Transformations | 214 |
| Data Sources | 214 |
| Data Flow Destinations | 216 |
| Transformation Types | 217 |
| Adding Data Transformations | 219 |
| Split Data Transformations | 221 |
| Translate Data Transformations | 223 |
| SSAS Data Transformations | 227 |
| Slowly Changing Dimension Transformation | 227 |
| Sample Data Transformations | 231 |
| Run Command Data Transformations | 231 |
| Enterprise Edition–Only Data Transformations | 232 |
| Using the Dynamic Package Configuration Wizard | 234 |
| Assigning SSIS Expressions | 236 |
| Summary | 236 |

| | |
|---|------------|
| ■ Chapter 9: Advanced SSIS | 237 |
| Understanding Package Execution | 237 |
| Data Viewers | 241 |
| Debugging SSIS Packages | 243 |
| Logging Execution Results | 245 |
| Error Handling | 248 |
| Event Handlers | 251 |
| Deploying SSIS Packages | 252 |
| SSIS Package Deployment Options | 253 |
| SSIS Package Execution Options | 256 |
| SSIS Package Security | 260 |
| Placing Checkpoints | 261 |
| Using Transactions in SSIS Packages | 262 |
| Data Profiling | 264 |
| Creating a Data Profile | 264 |
| Viewing a Data Profile | 267 |
| Summary | 270 |
| ■ Chapter 10: Reporting Tools | 271 |
| Using Excel Pivot Tables and Pivot Charts | 271 |
| Creating a Pivot Table | 271 |
| Creating a Pivot Chart | 274 |
| Publishing Your Workbook | 275 |
| Using SQL Server Reporting Services | 276 |
| SSRS Components | 276 |
| SSRS Reporting Samples | 277 |
| Building Your First SSRS Report | 277 |
| Running the Report Server Project Wizard | 278 |
| Designing the Query | 279 |
| Previewing and Designing Your Report | 284 |
| Publishing Your Report | 287 |
| Producing Reports with Report Builder | 290 |

| | |
|---|------------|
| Creating a Report Model | 290 |
| Creating a Dataset | 292 |
| Creating a Report | 293 |
| Summary | 299 |
| ■ Chapter 11: Data Mining with Excel | 301 |
| Exploring Excel 2010..... | 301 |
| The Excel Ribbon..... | 301 |
| KPI Support in Excel..... | 306 |
| Using Excel for Data Mining | 308 |
| Configuring Excel as a Data Mining Client | 309 |
| Using Excel as a Data Mining Client..... | 312 |
| Using the Data Preparation Group..... | 318 |
| Using the Data Modeling Group | 323 |
| Using the Accuracy and Validation Group | 324 |
| Summary | 328 |
| ■ Chapter 12: Introducing PowerPivot..... | 329 |
| The PowerPivot for Excel GUI..... | 329 |
| The PowerPivot Ribbon | 329 |
| The PowerPivot Designer | 332 |
| Using PowerPivot with Adventure Works..... | 336 |
| Importing Adventure Works Data | 336 |
| Enriching the Adventure Works Data | 338 |
| Using PowerPivot Data in Excel | 345 |
| Summary | 346 |
| ■ Chapter 13: Introduction to MDX..... | 347 |
| MDX Query Syntax | 347 |
| Understanding the Core Terminology..... | 348 |
| Learning the Basic Syntax | 350 |
| Writing Your First MDX Query | 352 |
| Discovering Members, Tuples, and Sets..... | 353 |

| | |
|--|------------|
| Calculated Members, Named Sets, and Script Commands | 355 |
| Adding MDX Objects to Your Cube | 356 |
| Using Calculated Measures..... | 359 |
| Working with Named Sets..... | 361 |
| Writing Script Commands | 362 |
| Common MDX Functions..... | 364 |
| Summary..... | 367 |
| ■ Chapter 14: Introduction to Data Mining..... | 369 |
| Defining SSAS Data Mining..... | 369 |
| Data-Mining Concepts..... | 372 |
| Architectural Considerations..... | 373 |
| Reviewing Data Mining Structures..... | 374 |
| Mining Structure Tab | 374 |
| Mining Models Tab..... | 375 |
| Mining Model Viewer Tab..... | 378 |
| Mining Accuracy Chart Tab | 381 |
| Mining Model Prediction Tab | 382 |
| Understanding and Using the Included Data Mining Algorithms..... | 383 |
| The Nine Algorithms..... | 384 |
| The Data Mining Wizard | 390 |
| Content and Datatypes..... | 392 |
| Processing Mining Models | 396 |
| Processing Methods..... | 396 |
| SSIS and Data Mining | 397 |
| Working with the DMX Language..... | 399 |
| Summary..... | 402 |
| ■ Appendix: The HIERARCHYID Datatype | 403 |
| Creating A HIERARCHYID Table | 403 |
| Adding Data to the Table..... | 404 |
| Displaying Hierarchical Data in SSMS | 405 |
| Index:..... | 407 |

About the Authors



■ **Guy Fouché** is a business intelligence and decision support system consultant in the Dallas, Texas area. Guy spends his evenings playing one of his eight trumpets and expanding his composition skills by using the current generation of music technologies. On the weekend, he puts as many miles as he can on his bright green Honda 600RR sport motorcycle. Guy and his wife Jodi enjoy taking nine-day trips in their Jeep 4×4, taking photographs and writing travelogs along the way. You can view their photography at <http://photography.fouche.ws>.



■ **Lynn Langit** is the founder and lead architect of WebFluent, which for the past six years has trained users and developers in building BI solutions. A holder of numerous Microsoft certifications, including MCT, MCITP, MCDBA, MCS.D.NET, MCSE, and MSF, she also has ten years of experience in business management. This unique background makes her particularly qualified to share her expertise in developing successful realworld BI solutions using SQL Server 2005. Lynn has recently joined Microsoft, working as a Developer Evangelist. She is based in the Southern California territory. For more information, read her blog at <http://blogs.msdn.com/SoCalDevGal>.

About the Technical Reviewer



■ **Michael Coles** is a Microsoft MVP (SQL) and consultant based in New York City. Michael has written several articles and books on a wide variety of SQL Server topics, including *Pro SQL Server 2008 XML* and the *Pro T-SQL 2008 Programmer's Guide*. He can be reached at www.sqlkings.com or www.sergeantsql.com.

Acknowledgments

I'd like to offer a huge thank you to everyone at Apress who has had input into these pages!