Microsoft

MCTS EXAM
70-448

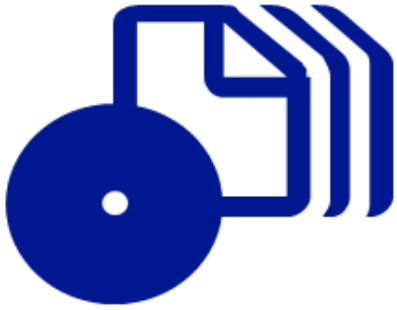
Microsoft® SQL Server® 2008– Business Intelligence Development and Maintenance

Erik Veerman,
Teo Lachev, and
Dejan Sarka of Solid
Quality Mentors

SELF-PACED

Training Kit

How to access your CD files



The print edition of this book includes a CD. To access the CD files, go to http://aka.ms/626362/files, and look for the Downloads tab.

Note: Use a desktop web browser, as files may not be accessible from all ereader devices.

Questions? Please contact: mspinput@microsoft.com

Microsoft Press

Exam 70-448: TS: Microsoft SQL Server 2008, Business Intelligence Development and Maintenance

OBJECTIVE	CHAPTER	LESSON
IMPLEMENTING AN SSIS SOLUTION		
Implement control flow.	Chapter 1 Chapter 2	Lesson 2 Lesson 1
Implement data flow.	Chapter 1 Chapter 2 Chapter 2 Chapter 2	Lesson 3 Lesson 1 Lesson 2 Lesson 3
Implement dynamic package behavior by using property expressions.	Chapter 3	Lesson 1
Implement package logic by using variables.	Chapter 1	Lesson 2
Implement package configurations.	Chapter 3	Lesson 1
Implement auditing, logging, and event handling.	Chapter 2 Chapter 2	Lesson 2 Lesson 3
Extend SSIS packages by using .NET code.	Chapter 1	Lesson 2
CONFIGURING, DEPLOYING, AND MAINTAINING SSIS		
Install and maintain SSIS components.	Chapter 1 Chapter 2 Chapter 4	Lesson 1 Lesson 1 Lesson 1
Deploy an SSIS solution.	Chapter 3	Lesson 2
Manage SSIS package execution.	Chapter 4	Lesson 2
Configure SSIS security settings.	Chapter 4	Lesson 1
Identify and resolve issues related to SSIS solution deployment.	Chapter 1 Chapter 2 Chapter 2 Chapter 3	Lesson 1 Lesson 2 Lesson 3 Lesson 2
IMPLEMENTING AN SSAS SOLUTION		
Implement dimensions in a cube.	Chapter 5 Chapter 5 Chapter 6	Lesson 2 Lesson 3 Lesson 1
Implement measures in a cube.	Chapter 5 Chapter 5	Lesson 2 Lesson 4
Implement a data source view.	Chapter 5	Lesson 1
Configure dimension usage in a cube.	Chapter 6	Lesson 1
Implement custom logic in a cube by using MDX.	Chapter 6 Chapter 6	Lesson 2 Lesson 3
Implement data mining.	Chapter 9 Chapter 9 Chapter 9 Chapter 9	Lesson 1 Lesson 2 Lesson 3 Lesson 4
Implement storage design in a cube.	Chapter 7	Lesson 1

CONFIGURING, DEPLOYING, AND MAINTAINING SSAS		
Configure permissions and roles in SSAS.	Chapter 8	Lesson 1
Deploy SSAS databases and objects.	Chapter 7	Lesson 2
Install and maintain an SSAS instance.	Chapter 8 Chapter 8	Lesson 2 Lesson 3
Diagnose and resolve performance issues.	Chapter 8	Lesson 3
Implement processing options.	Chapter 7	Lesson 3
IMPLEMENTING AN SSRS SOLUTION		
Implement report data sources and datasets.	Chapter 10 Chapter 12	Lesson 1 Lesson 3
Implement a report layout.	Chapter 10 Chapter 10 Chapter 11	Lesson 1 Lesson 3 Lesson 2
Extend an SSRS solution by using code.	Chapter 11	Lesson 4
Create an SSRS report by using an SSAS data source.	Chapter 10	Lesson 2
Implement report parameters.	Chapter 11 Chapter 11	Lesson 1 Lesson 2
Implement interactivity in a report.	Chapter 10 Chapter 11	Lesson 3 Lesson 2
Implement report items.	Chapter 10 Chapter 10 Chapter 11	Lesson 1 Lesson 4 Lesson 2
Embed SSRS reports in custom applications.	Chapter 11	Lesson 3
CONFIGURING, DEPLOYING, AND MAINTAINING SSRS		
Configure report execution and delivery.	Chapter 12 Chapter 12	Lesson 2 Lesson 3
Install and configure SSRS instances.	Chapter 13 Chapter 13	Lesson 1 Lesson 2
Configure authentication and authorization for a reporting solution.	Chapter 12	Lesson 2
Deploy an SSRS solution.	Chapter 11	Lesson 3
Configure SSRS availability.	Chapter 13 Chapter 13	Lesson 1 Lesson 2

Exam Objectives The exam objectives listed here are current as of this book's publication date. Exam objectives are subject to change at any time without prior notice and at Microsoft's sole discretion. Please visit the Microsoft Learning Web site for the most current listing of exam objectives: http://www.microsoft.com/learning/en/us/exams/70-448.mspx.

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Control Number: 2009920806

Printed and bound in the United States of America.

ISBN: 978-0-7356-2636-2

6 7 8 9 10 11 12 13 14 QG 7 6 5 4 3 2

Distributed in Canada by H.B. Fenn and Company Ltd.

A CIP catalogue record for this book is available from the British Library.

Microsoft Press books are available through booksellers and distributors worldwide. For further information about international editions, contact your local Microsoft Corporation office or contact Microsoft Press International directly at fax (425) 936-7329. Visit our Web site at www.microsoft.com/mspress. Send comments to tkinput@microsoft.com.

Acquisitions Editor: Ken Jones
Developmental Editor: Laura Sackerman
Project Editor: Maureen Zimmerman
Editorial Production: Online Training Solutions, Inc.
Technical Reviewer: Rozanne Murphy Whalen; Technical Review services provided by Content Master, a member of CM Group, Ltd.
Cover: Tom Draper Design

Body Part No. X15-52846

[2012-11-30]

To my children . . . Meg, Nate, Kate, and Caleb.

—Erik Veerman

To my family, for tolerating my absence during the writing of this book.

—Teo Lachev

To my son.

—Dejan Sarka

Acknowledgments

First, thank you to Teo and Dejan for their excellent work and dedication to the effort to make this book project a success. Also thanks to my many clients who have provided real-life BI experiences—both the good and the ugly, but I won't tell you which ones! Next, there's no better place to work than with Solid Quality Mentors—a special thanks to Douglas McDowell and Brian Moran, who make our firm one of the best in the world. My book dedication is to my children, but it is my wonderful wife, Amy, who makes this all possible. Thank you for your patience. *Sola gratia, sola fide, solo Christos*.

—Erik Veerman

I would like to thank Erik Veerman and Solid Quality Mentors for entrusting me to write the Analysis Services part of *MCTS Self-Paced Training Kit (Exam 70-445): Microsoft SQL Server 2005 Business Intelligence—Implementation and Maintenance* and this book. Over the past several years, I've been privileged to personally know and work with bright and talented developers who have contributed tremendously to the success of the Microsoft Business Intelligence Platform. Erik and Dejan are two of the best. Their professionalism, experience, and dedication have shown through this book again. Thank you for making this project a smooth ride!

—Teo Lachev

It was a great pleasure to work with Erik and Teo again. In addition, thanks to all friends from Solid Quality Mentors.

—Dejan Sarka

Contents at a Glance

Contents

Introduction

This Training Kit is designed for business intelligence (BI) developers and administrators who plan to take the Microsoft Certified Technology Specialist (MCTS) Exam 70-448, *Microsoft SQL Server 2008, Business Intelligence Development and Maintenance*. The primary objective of this exam is to certify that BI developers and administrators know how to develop and maintain solutions built on the Microsoft SQL Server 2008 BI platform, which includes SQL Server Integration Services (SSIS), SQL Server Analysis Services (SSAS), and SQL Server Reporting Services (SSRS). We assume that before you begin using this Training Kit, you have experience developing or implementing BI solutions. We also assume that you have experience managing or supporting BI project security, deployment, and maintenance. The Preparation Guide for Exam 70-448 is available from *http://www.microsoft.com/learning/exams/70-448.mspx*. The practice exercises in this Training Kit require you to use Microsoft SQL Server 2008 Enterprise or Microsoft SQL Server 2008 Developer. A 180-day evaluation edition of SQL Server 2008 Enterprise is included on this book's SQL Server 2008 evaluation DVD. If you do not have access to this software, you can download a 180-day trial of SQL Server 2008 from *http://www.microsoft.com/sqlserver/2008/en/us/trial-software.aspx*. You can also consider purchasing SQL Server 2008 Development, which contains all of the required features.

By using this Training Kit, you will learn how to:

- Install and configure the SQL Server 2008 BI components.
- Work with the design and management tools in SQL Server 2008 for BI.
- Develop and deploy SSIS projects.
- Secure, manage, and troubleshoot SSIS packages.
- Develop and deploy SSAS solutions.
- Secure SSAS cubes and dimensions.
- Implement, configure, and deploy SSRS reports.
- Manage and secure SSRS report servers.

Hardware Requirements

We recommend that you use a test workstation, test server, or staging server to complete the exercises in each practice. However, it would be beneficial for you to have access to production-ready data in your organization. If you need to set up a workstation to complete the practice exercises, the minimum 32-bit system (X86) requirements for installing SQL Server 2008 are:

- A computer with a 1-GHz Pentium III compatible or faster processor (2 GHz or faster recommended).
- 512 MB of RAM or more (2 GB or higher recommended).
- 2.1-GB free hard disk space for the SQL Server installation files and samples (which include all of the BI services, client components, developer and management tools, sample databases and projects, and online help files).
- A DVD-ROM drive for installing SQL Server 2008 from the evaluation software DVD.
- A Super VGA (1024 × 768) or higher resolution video adapter and monitor.
- A keyboard and Microsoft mouse, or compatible pointing device.

For detailed SQL Server 2008 hardware requirements, see *http://technet.microsoft.com/en-us/library/ms143506.aspx*. You can also install SQL Server 2008 on a virtual machine instead of on standard computer hardware by using the virtual machine software Virtual PC 2007, Virtual Server 2005 R2, Hyper-V, or third-party virtual machine software. To download an evaluation of Virtual Server 2005 R2, go to *http://www.microsoft.com/virtualserver*. For more information about Hyper-V, go to *http://www.microsoft.com/hyperv*. To download Virtual PC for free, go to *http://www.microsoft.com/windows/products/winfamily/virtualpc/default.mspx*.

Software Requirements

Note that you will need SQL Server 2008 installed with the BI components, tools, and samples in order to complete the practices included with each chapter. Although these products can be installed on a production server, it is not recommended that you use a production installation for this Training Kit. Instead, install these products and execute the practices on a single development computer. The following software is required to complete the practice exercises:

- **A compatible operating system** SQL Server 2008 can be installed on many versions of Windows server and desktop operating systems, including Windows XP (with Service Pack 2 [SP2] or later), Windows Server 2003 (with SP2), Windows Vista, and Windows Server 2008. See *http://technet.microsoft.com/en-us/library/ms143506.aspx* to help you choose a compatible SQL Server 2008 version.

  In general, SQL Server 2008 Enterprise can be installed on many of the server operating system products (such as Windows Server 2003 SP2 or Windows Server 2008), but it cannot be installed on the desktop operating systems.

SQL Server 2008 Developer can be installed on the same Windows Server editions that the Enterprise edition can be installed on, and it can also be installed on the desktop operating systems, such as Windows XP SP2 and Windows Vista.

- **SQL Server 2008** A 180-day evaluation of SQL Server Enterprise is included on the evaluation software DVD. A 180-day evaluation of SQL Server 2008 is also available as a free download from the Microsoft Developer Network (MSDN) Web site at *http://www.microsoft.com/sqlserver/2008/en/us/trial-software.aspx*. Instructions for installing the BI components of SQL Server 2008 are included in the next section.
- **Microsoft .NET Framework 3.5** This is required to be installed before the SQL Server 2008 installation setup process can be initiated. This prerequisite is available with the installation files on the SQL Server 2008 evaluation DVD.
- **Microsoft Visual Studio 2008 (optional)** You use Visual Studio 2008 Standard or Visual Studio 2008 Professional installed with the Microsoft Visual Basic .NET library to complete the practice exercises for Chapter 11. You must also install Visual Studio 2008 SP1 (or later).

  A 90-day trial version of Visual Studio 2008 Professional is available at *http://www.microsoft.com/downloads/details.aspx?FamilyID=83c3a1ec-ed72-4a79-8961-25635db0192b&displaylang=en*. You can download Visual Studio 2008 SP1 by going to *http://www.microsoft.com/downloads/details.aspx?FamilyId=FBEE1648-7106-44A7-9649-6D9F6D58056E&displaylang=en*.

  You should install Visual Studio 2008 only when you are ready to start Chapter 11, because doing so changes the menu options you see in Business Intelligence Development Studio (BIDS). For example, to create a new project in BIDS when you have Visual Studio installed, you choose File and then New Project. In contrast, to create a new project in BIDS when you do not have Visual Studio installed, you choose File, New, and then Project.

NOTE **USING BIDS AND VISUAL STUDIO TOGETHER**

With the exception of Chapter 11, this book was written under the assumption that you do not have Visual Studio installed. If you already have Visual Studio installed, you will find that your menu options in BIDS will differ slightly from what is written in the book for the procedures and practice exercises.

- **The sample relational databases named AdventureWorks2008 and AdventureWorksDW2008** These are available for download here: *http://www.microsoftpresstore.com/title/9780735626362*.

- **The sample SSAS database named Adventure Works DW 2008** The sample SSAS database is available with the SQL Server 2008 product samples for download here: *http://www.microsoftpressstore.com/title/9780735626362*.
- **The AdventureWorks report samples** The AdventureWorks report samples consist of report definition files for SQL Server Reporting Services that reference the AdventureWorks 2008 databases. The samples are available for download here: *http://* www.microsoftpressstore.com/title/9780735626362. After you download and install the sample reports, you must deploy the reports within BIDS.

For detailed hardware requirements, see *http://technet.microsoft.com/en-us/library/ms143506.aspx*. It should also be noted that Internet Information Services (IIS) is not required for Reporting Services 2008 installation.

Installing SQL Server 2008

Either SQL Server 2008 Enterprise or SQL Server 2008 Developer is required to run the code samples and practices provided in this book. A 180-day evaluation edition is available on this book's SQL Server 2008 evaluation DVD. Alternatively, a free 180-day evaluation edition of SQL Server 2008 Enterprise is available for download from *http://www.microsoft.com/sqlserver/2008/en/us/trial-software.aspx.* This version can be installed on both the server operating system and desktop operating system with which SQL Server 2008 is compatible.

The SQL Server 2008 platform includes the core Database Engine, BI components, and tools to support development and administration. SQL Server 2008 is available in different editions and languages. The editions include feature subsets intended for a variety of purposes and applications. The primary editions of SQL Server 2008 are:

- **SQL Server 2008 Enterprise** Includes the full features of SQL Server 2008 and provides enterprise performance and capabilities.
- **SQL Server 2008 Developer** Includes the full features of SQL Server 2008 and can be used for development.
- **SQL Server Standard** Includes the core functionality of SQL Server 2008 but does not contain the scalability options and advanced capabilities that SQL Server 2008 Enterprise and SQL Server 2008 Developer contain.

- **SQL Server 2008 Workgroup** Contains the core database components but is limited in functionality, with only a small subset of BI features.
- **SQL Server 2008 Web** Contains the core database components with limited functionality and also includes Reporting Services with limited functionality. This version is intended for Web applications and workloads.
- **SQL Server 2008 Express** Freely distributable lightweight edition of SQL Server 2008. This version has limitations but can be used for development and to embed in applications.
- **SQL Server 2008 Compact** The lightweight mobile version of SQL Server 2008 Database Engine.

For a comprehensive description of each edition's capabilities, see *http://www.microsoft.com/Sqlserver/2008/en/us/editions.aspx*. The focus of this Training Kit is the implementation and management of the BI components found in the Enterprise, Developer, and Standard editions of SQL Server 2008.

EXPLORING THE SQL SERVER INSTALLATION CENTER

All the features of SQL Server 2008 are available on the SQL Server 2008 evaluation DVD and can be installed on the same server. You can also install the features on separate servers if that works best within your BI architecture.

The installation of SQL Server 2008 components are launched through the SQL Server Installation Center, which runs automatically when the evaluation DVD is inserted and provides installation and setup resources for stand-alone installations, upgrades, failover cluster installs, tool installs, and so on. Figure I-1 shows the Installation page of the SQL Server Installation Center.

For a new installation or to modify an existing installation, click the New SQL Server Stand-alone Installation Or Add Features To An Existing Installation link on the Installation page.

IMPORTANT **LICENSING SQL SERVER 2008**

A SQL Server 2008 license is required for each server on which you install any of the server components; however, running multiple components of SQL Server 2008 on the same server requires only a single license. You can find complete licensing details at *http://www.microsoft.com/sqlserver/2008/en/us/licensing.aspx*.

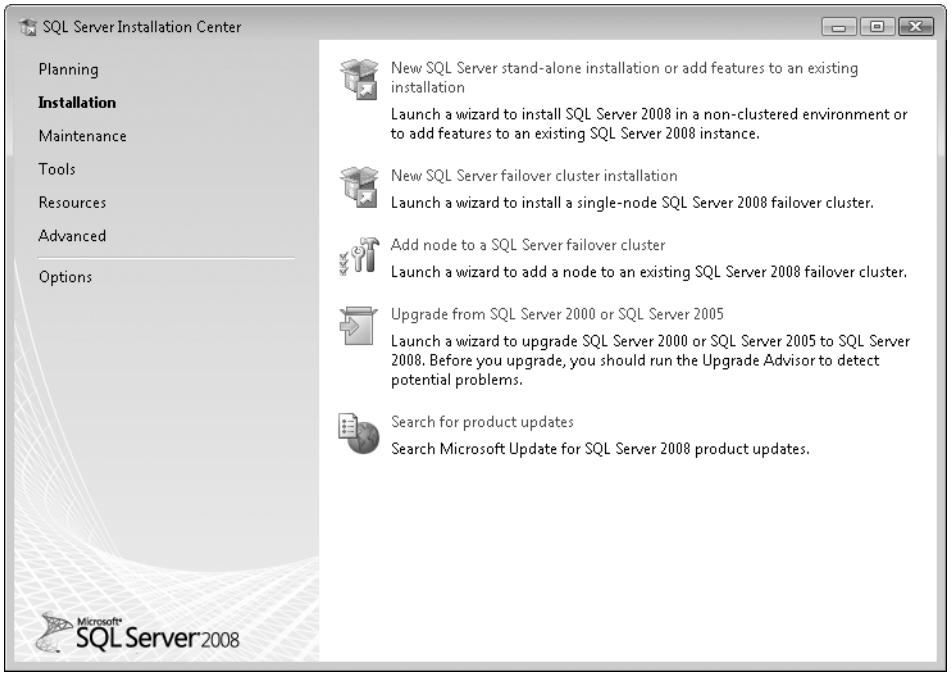


FIGURE I-1 The Installation page of the SQL Server Installation Center provides links that launch the SQL Server 2008 installation.

Selecting Installation Components

The first step of the installation will check the minimum hardware requirements and will install Microsoft .NET Framework 3.5 and SQL Server Native Client. These prerequisites are included on the SQL Server 2008 evaluation DVD. Furthermore, during the initial setup, the installer will identify other required supporting applications and Windows components you might need.

The SQL Server installation process will then scan your computer for the required configuration. The System Configuration Check results will indicate whether configuration changes need to be made before the installation proceeds. If any configurations are not correct, Setup will block the installation of SQL Server 2008. After the prerequisites and configuration check, you will be able to select the features for installation on the Feature Selection page of the SQL Server 2008 Setup Wizard. For a complete installation, select all the components, as shown in Figure I-2.

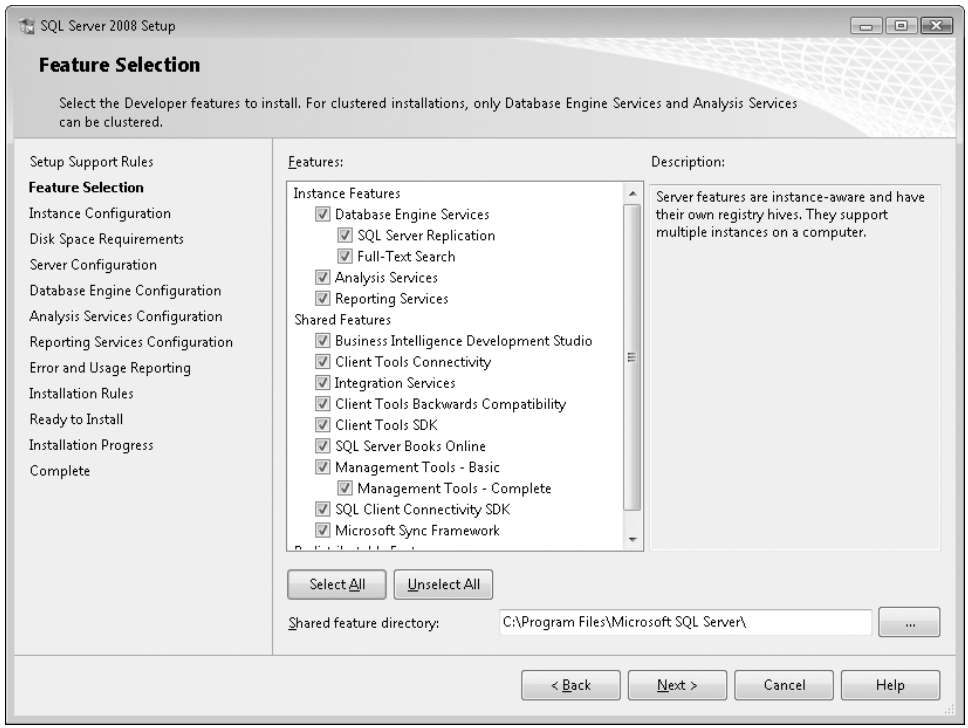


FIGURE I-2 On the Feature Selection page, select all the components for a complete SQL Server installation.

Choosing Installation Details

After the component selections are complete, the next installation steps are determining the installation details, such as selecting the instance name, setting the program and data file locations, and identifying the appropriate security accounts.

SPECIFYING AN INSTANCE NAME

The first selection you will be prompted to make will determine the instance name. Several components of SQL Server 2008 can be installed on the same computer multiple times. Each time the same component is installed, it needs a new instance name for that installation. Instances apply to the Database Engine, Analysis Services, and Reporting Services.

- Choosing the Default Instance means that the installation components that you selected will be installed with no name.
- Alternatively, you can name the new installation instance by using the Named Instance option.

Figure I-3 shows the Instance Configuration page of the SQL Server 2008 Setup Wizard with the Default Instance option selected.

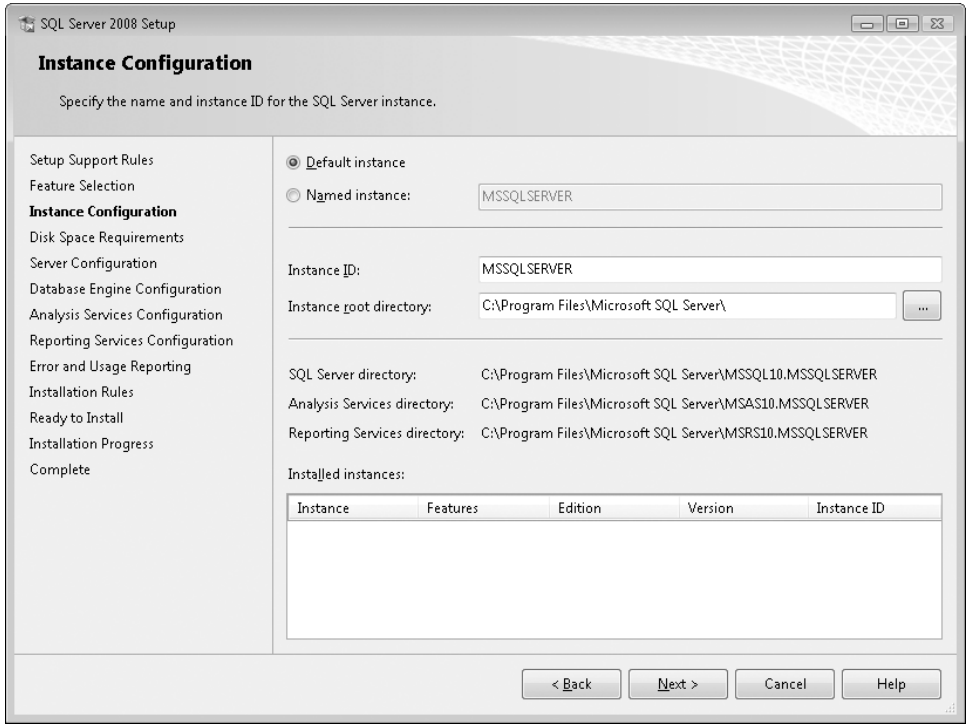


FIGURE I-3 The Instance Configuration page allows you to either choose the Default Instance or create a Named Instance for all the installation components.

When you choose the default instance, the connection strings to access the servers need to contain only the server name and not the named instance extension. Having multiple named instances also allows you to install different versions and editions of SQL Server on the same physical computer. For example, you can have multiple installations of SSAS on the same physical computer, each at different service pack levels.

CUSTOMIZING SERVICE ACCOUNTS

The Server Configuration page allows you to customize the security accounts that each service will use to run. On the Service Accounts tab of the Server Configuration page, shown in Figure I-4, you can indicate the service account to be used. For each service, the Account Name can use a local account or domain account where you specify the account password. Alternatively, you can also choose the LOCAL SYSTEM or Network Service account and a password is not required. However, access to local and external domain resources might be limited.

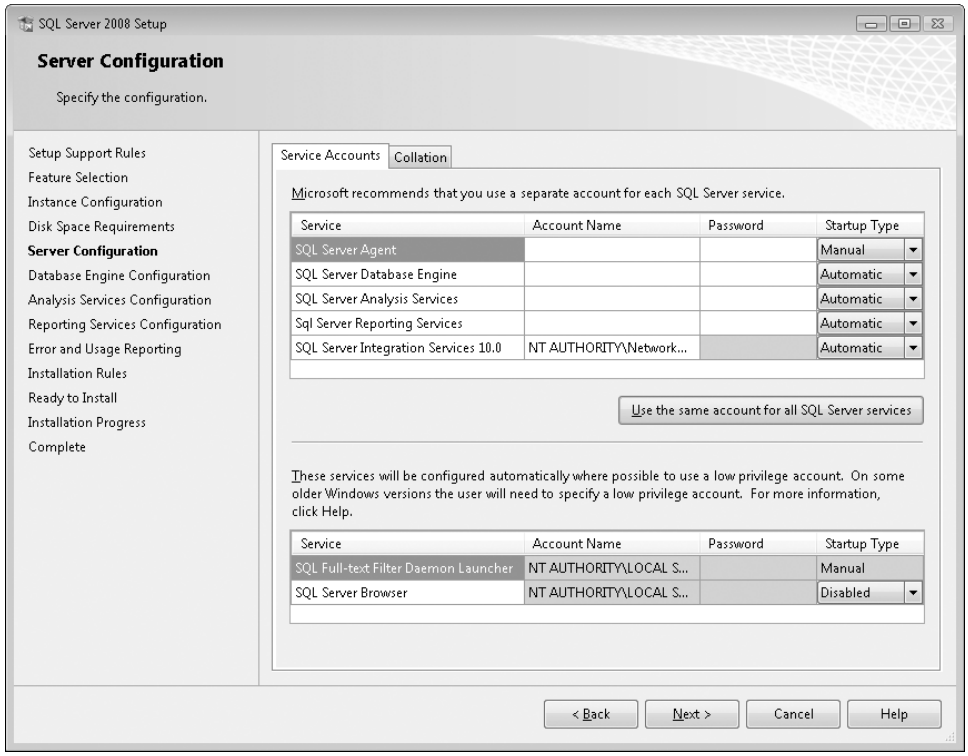


FIGURE I-4 On the Service Accounts tab of the Server Configuration page, you can define the security accounts that are used when each service starts.

To build a test system, such as for the purposes of this Training Kit, you can set all the services to use the LOCAL SYSTEM account. Click the Use The Same Account For All SQL Server Services button, and then choose the LOCAL SYSTEM account.

The Collation tab defines how the Database Engine handles data sorting based on locale settings, case sensitivity, and binary order.

The Database Engine collation settings can be defined independently from Analysis Services collation settings. To define separate collations, select the Customize For Each Service Account check box, and then change the value of the Service drop-down list for each service on the Server Configuration page.

SETTING THE AUTHENTICATION MODE

The Authentication Mode setting is specific to the SQL Server Database Engine and defines the way in which users are able to log on to SQL Server. Figure I-5 shows the Database Engine Configuration settings.

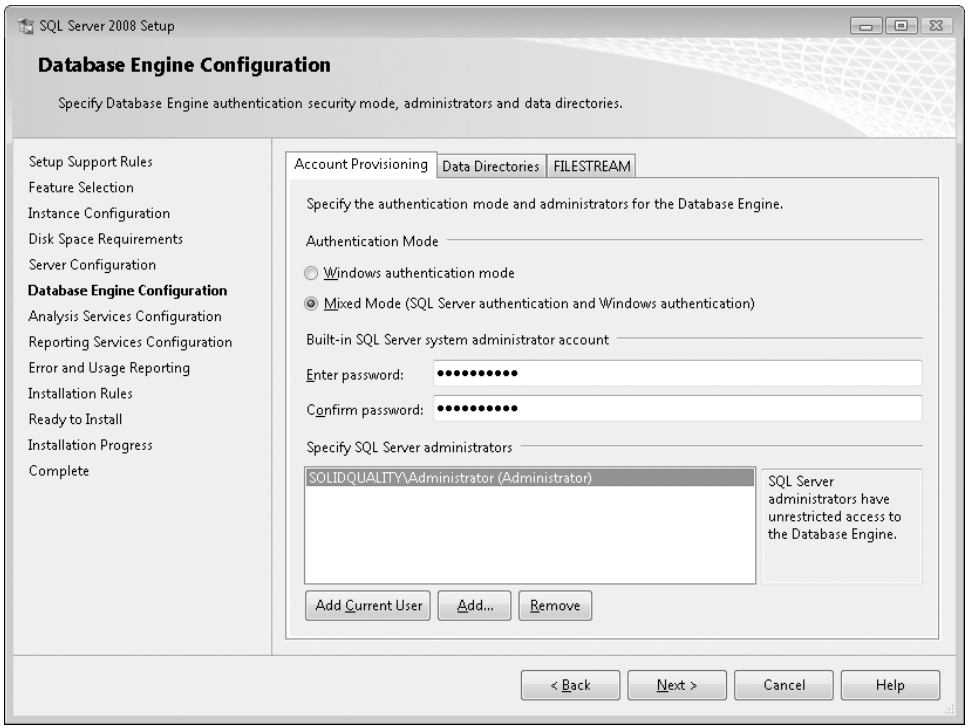


FIGURE I-5 The Database Engine Configuration page is used to set the SQL Server security authentication mode.

- The Windows Authentication Mode option specifies that a user can connect only with a local computer account or domain account.
- The Mixed Mode option allows users to connect with Windows Authentication or with authentication defined in SQL Server.

Note that you can change the Authentication Mode setting after installation by using the Server Properties dialog box in SQL Server Management Studio.

Click the Add Current User button or the Add button to add your personal account as a SQL Server administrator or another account as an administrator. Furthermore, the folders for storing files such as log files, data files, backup files, and temp folders can be set on the Data Directories tab. Filestream can be enabled on the FILESTREAM tab for accessing unstructured file data through SQL Server.

CONFIGURING ANALYSIS SERVICES

The installation settings for Analysis Services include defining the administrator accounts and setting folders for the data, backup, and log files, as shown in Figure I-6.

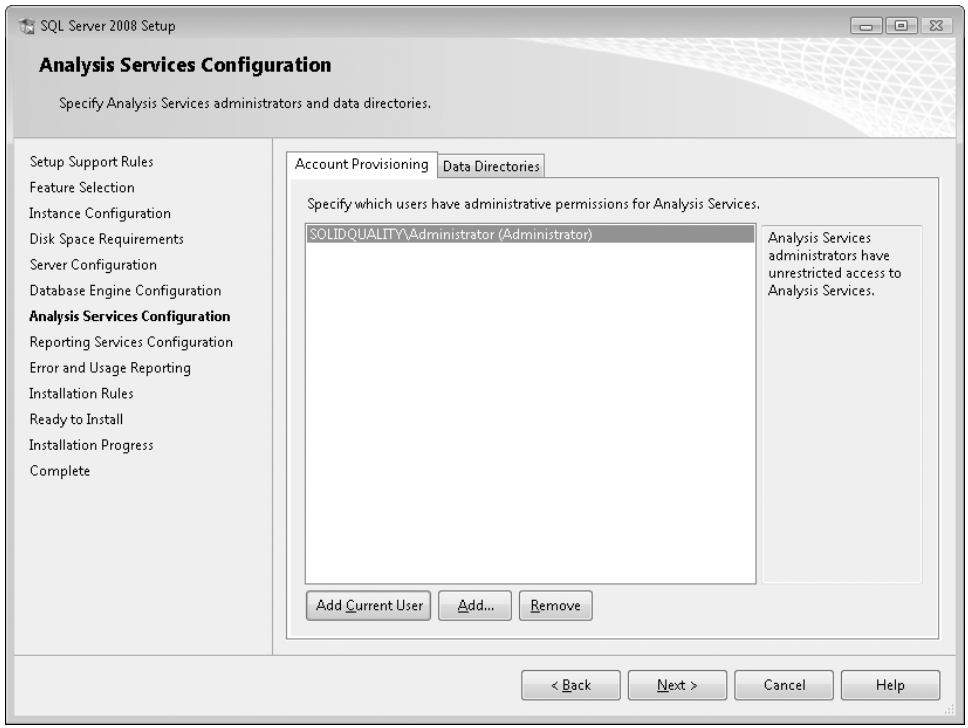


FIGURE I-6 The Analysis Services administrator accounts and data folders can be defined during setup.

CONFIGURING REPORTING SERVICES

For SQL Server 2008 implementations that include Reporting Services, during the installation, you can either choose the default SSRS configuration or choose to configure the SSRS service later, but you cannot customize the SSRS installation settings. In other words, you can choose to have Reporting Services configured with the default configurations, or you can have Setup install Reporting Services but leave it unconfigured and then configure it after installation. Chapter 13, "Configuring and Administering the SSRS Server," reviews the custom configuration for Reporting Services. Figure I-7 shows the Reporting Services Configuration page of the SQL Server 2008 Setup Wizard.

If SharePoint Services is installed, Reporting Services can also be installed in SharePoint integrated mode so that you can administer the report server and users can access reports through SharePoint.

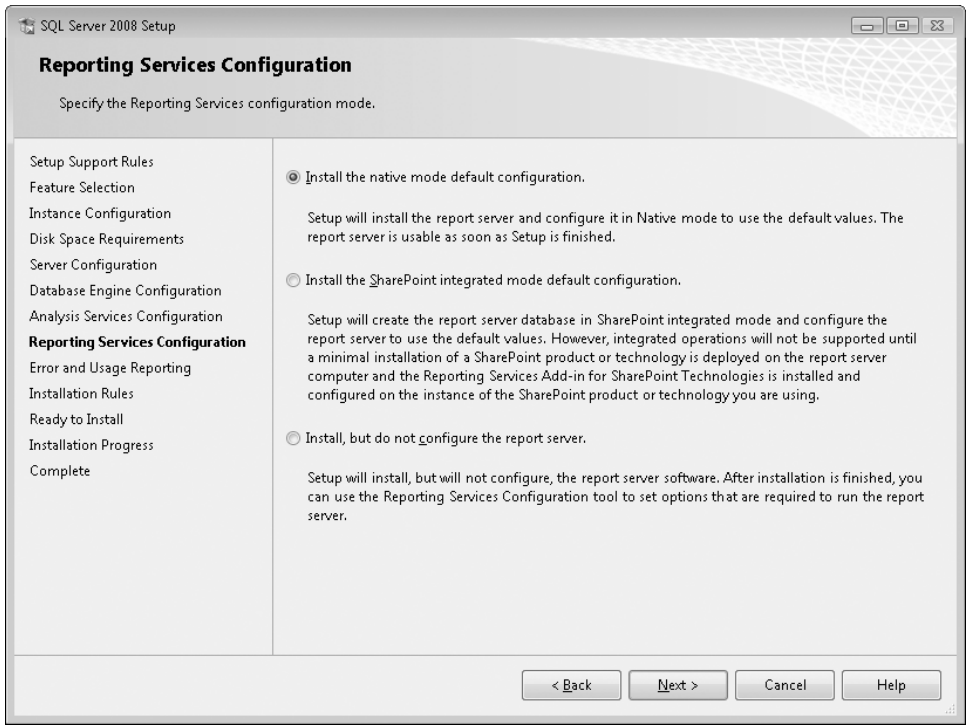


FIGURE I-7 Reporting Services can be installed in native mode, installed with SharePoint integrated mode, or installed but not configured.

COMPLETING THE INSTALLATION

On the remaining wizard pages, you can specify whether to send errors to Microsoft, perform final installation checks, and to confirm the installation detail summary. Clicking Install on the Ready To Install page will run the file copy and installation process until the setup process is complete.

Using the SQL Server Evaluation DVD and the Companion CD

A companion CD and a SQL Server 2008 evaluation DVD are included with this Training Kit. The companion CD contains the following:

- **Practice tests** You can practice for the 70-448 certification exam by using tests created from a pool of 200 realistic exam questions. These questions give you many different practice exams to ensure that you are prepared to take the real test.

- **Chapter practice exercises** Many chapters in this book include sample files associated with the practice exercises at the end of every lesson. Most exercises have a project or solution that you can use to complete the exercise and a version of the completed exercise for your review. To install the sample files on your hard disk, run Setup.exe from the Practice folder on the companion CD. The default installation folder is C:\Users*username*\Documents\Microsoft Press\MCTS Training Kit 70-448\Source\. Within the Source folder, you will find a separate folder corresponding to each chapter in the book.
- **An eBook** An electronic version of this book (an eBook) is included for times when you do not want to carry the printed book with you. The eBook is in Portable Document Format (PDF), and you can view it by using Adobe Acrobat or Adobe Acrobat Reader, available from *http://www.adobe.com*.
- **Sample chapters** Sample chapters from related Microsoft Press titles are offered on the CD. These chapters are in PDF format.
- **Glossary** A glossary of terms used in this book is included on the companion CD. The glossary is in PDF format, viewable by using Adobe Acrobat or Adobe Acrobat Reader.

Digital Content for Digital Book Readers: If you bought a digital-only edition of this book, you can enjoy select content from the print edition's companion CD.
Visit **http://www.microsoftpressstore.com/title/9780735626362.** to get your downloadable content. This content is always up-to-date and available to all readers.

Installing the Practice Tests

To install the practice test software from the companion CD on your hard disk, perform the following steps:

1. Insert the companion CD into your CD drive, and then accept the license agreement. A CD menu appears.

> ***NOTE*** **IF THE CD MENU DOES NOT APPEAR**
>
> **If the CD menu or the license agreement does not appear, AutoRun might be disabled on your computer. Refer to the Readme.txt file on the CD for alternative installation instructions.**

2. Select the Practice Tests item, and then follow the instructions on the screen and then follow the instructions in the Microsoft Press Training Kit Exam Prep Suite 70-448 Wizard.

Using the Practice Tests

To start the practice test software, follow these steps:

1. Click Start, select All Programs, select Microsoft Press Training Kit Exam Prep, and then select Microsoft Press Training Kit Exam Prep again. A window appears that shows all the Microsoft Press training kit exam prep suites that are installed on your computer.
2. Double-click the practice test that you want to use.

CHOOSING PRACTICE TEST OPTIONS

When you start a practice test, you choose whether to take the test in Certification Mode, Study Mode, or Custom Mode.

- **Certification Mode** Closely resembles the experience of taking a certification exam. The test has a set number of questions, it is timed, and you cannot pause and restart the timer.
- **Study Mode** Creates an untimed test in which you can review the correct answers and the explanations after you answer each question.
- **Custom Mode** Gives you full control over the test options so that you can customize them to suit your needs. You can click OK to accept the defaults, or you can set the number of questions you want to answer, define the way the practice test software works, choose the exam objectives to which you want the questions to relate, and indicate whether you want your lesson review to be timed. If you are retaking a test, you can indicate whether you want to see all the questions again or only those questions you previously missed or did not answer.

In all modes, the user interface you see when taking the test is essentially the same, but depending on the mode, different options will be enabled or disabled.

After you click OK, your practice test starts.

- To take the test, answer the questions, and then use the Next, Previous, and Go To buttons to move from question to question.
- After you answer an individual question, to see which answers are correct and to see an explanation of each correct answer, click Explanation.
- If you would rather wait until the end of the test to see how you did, answer all the questions, and then click Score Test. You will see a summary of the exam objectives you chose, the percentage of questions you answered correctly overall, and the percentage of questions you answered correctly for each objective. You can print a copy of your test, review your answers, or retake the test.

When you review your answer to an individual practice test question, a "References" section lists the places in the Training Kit in which you can find the information that relates to that question and provides links to other sources of information. After you click Test Results

to score your entire practice test, you can click the Learning Plan tab to see a list of references for every objective.

Uninstalling the Practice Tests

To uninstall the practice test software for a Training Kit, use the Add Or Remove Programs option in Windows Control Panel.

System Requirements for the Companion CD

To use the companion CD, you need a computer running Windows Server 2008, Windows Vista, Windows Server 2003, or Windows XP Professional. The computer must meet the following minimum requirements:

- 1 GHz 32-bit (x86) or 64-bit (x64) processor (depending on the minimum requirements of the operating system)
- 1 GB of system memory (depending on the minimum requirements of the operating system)
- A hard disk partition with at least 1 GB of available space
- A monitor capable of at least 800 x 600 display resolution
- A keyboard
- A mouse or other pointing device
- An optical drive capable of reading CDs

The computer must also have the following software:

- A Web browser such as Windows Internet Explorer 7 or later
- An application that can display PDF files, such as Adobe Acrobat Reader, which can be downloaded at *http://www.adobe.com/reader*

These requirements support the use of the companion CD. To perform the practice exercises in this training kit, you need additional hardware and software. See the preceding sections for detailed requirements.

Microsoft Certified Professional Program

The Microsoft certifications provide the best method to prove your command of current Microsoft products and technologies. The exams and corresponding certifications are developed to validate your mastery of critical competencies as you design and develop or implement and support solutions with Microsoft products and technologies. Computer professionals who become Microsoft-certified are recognized as experts and are sought after

industry-wide. Certification brings a variety of benefits to the individual and to employers and organizations.

> ***NOTE*** **THE MICROSOFT CERTIFICATIONS**
>
> **For a full list of Microsoft certifications, go to *http://www.microsoft.com/learning/mcp/default.asp*.**

Technical Support

Every effort has been made to ensure the accuracy of this book and the contents of the companion CD. If you have comments, questions, or ideas regarding this book or the companion CD, please send them to Microsoft Press by using either of the following methods:

E-mail:

- tkinput@microsoft.com

Postal Mail:

- *Microsoft Press*

Attn: MCTS Self-Paced Training Kit (Exam 70-448): Microsoft SQL Server 2008—Business Intelligence Development and Maintenance *Editor*
One Microsoft Way
Redmond, WA 98052-6399

For additional support information regarding this book and the companion CD (including answers to commonly asked questions about installation and use), visit the Microsoft Press Technical Support Web site at *http:/www.microsoft.com/learning/support/books*. To connect directly to the Microsoft Knowledge Base and enter a query, visit *http://support.microsoft.com/search*. For support information regarding Microsoft software, please visit *http://support.microsoft.com*.

Evaluation Edition Software Support

The 180-day evaluation edition software provided with this Training Kit is not the full retail product and is provided only for the purposes of training and evaluation. Microsoft and Microsoft Technical Support do not support this evaluation edition.

Information about any issues relating to the use of this evaluation edition with this Training Kit is posted to the Support section of the Microsoft Press Web site at *http://www.microsoft.com/learning/support/books*. For information about ordering the full version of any Microsoft software, please call Microsoft Sales at (800) 426-9400 or visit the Microsoft Web site at *http://www.microsoft.com*.

CHAPTER 1

Developing SSIS Packages

A *package* is the core object within SQL Server Integration Services (SSIS) that contains the business logic to handle workflow and data processing. You use SSIS packages to move data from sources to destinations and to handle the timing precedence of when data is processed. You can create packages by using the SQL Server Import And Export Wizard in SQL Server Management Studio (SSMS) or by using the SSIS Designer in the Business Intelligence Development Studio (BIDS). This chapter looks at creating and defining packages in SSIS and using the main components of the control flow and data flow objects with sources and destinations.

SSIS is designed for many data integration and processing applications. One of those applications is the processing of data into a data mart or data warehouse, where data is used exclusively for business intelligence (BI) analytics and reporting. Although many businesses use SSIS for BI, there are many other applications of SSIS. For example, many organizations use SSIS to move data from legacy systems into new systems during application migrations, to integrate data from multiple systems by passing data back and forth, to extract data for sending to vendors or partners, to cleanse data, to import data from vendors or partners—the list goes on. Because this Training Kit focuses on BI, part of the SSIS content and lessons cover using SSIS for data warehouse extraction, transformation, and loading (ETL), but the SSIS chapters and lessons also explain how to take advantage of SSIS for other purposes.

This initial chapter explains how to create SSIS packages and defines the basic objects contained in the control flow and data flow. Later chapters describe the advanced features, deployment, and implementation details of SSIS.

Exam objectives in this chapter:

- Implement control flow.
- Implement data flow.
- Implement package logic by using variables.
- Extend SSIS packages by using .NET code.
- Identify and resolve issues related to SSIS solution deployment.
- Install and maintain SSIS components.

Before You Begin

To complete this chapter, you must have:

- Knowledge of Microsoft SQL Server 2008, including SSIS features and components.
- Experience working with SQL Server Business Intelligence Development Studio (BIDS) projects and solutions.
- Experience working in SQL Server Management Studio (SSMS).
- The AdventureWorks2008 and AdventureWorksDW2008 sample databases installed. You can download these databases here: *http://go.microsoft.com/FWLink/?Linkid=272925*.

Lesson 1: Creating SSIS Packages and Data Sources

Estimated lesson time: 50 minutes

The core object within SSIS is a *package*. A package contains the business logic to handle the data extraction, manipulation, and transformation tasks needed to move data to destinations. Packages also contain workflow elements to help process data. These workflow elements might involve running a stored procedure, moving a file from an FTP server to a destination folder on your server, or sending an e-mail message when an error occurs. When you execute a package, the logic within performs the designed steps.

Packages also contain connections to data sources and data destinations. You set up these connections to connect to different external systems such as databases, files, File Transfer Protocol (FTP) servers, Simple Mail Transfer Protocol (SMTP) servers, and so on. Connections are used for the SSIS data processing engine (called the *data flow*) as well as the workflow engine (called the *control flow*).

Creating SSIS Packages

The first step in getting started with SSIS is to create a package. You can accomplish this in one of two ways:

- By using the built-in Import And Export Wizard in SQL Server 2008, which asks you about moving data from a source to a destination and then automatically generates an SSIS package. After you create a package in the wizard, you can execute it immediately, schedule it, or associate it with an SSIS project.
- By explicitly creating a package inside an SSIS project in BIDS. BIDS in SQL Server 2008 uses the Microsoft Visual Studio 2008 interface with specific templates installed to create BI objects such as SSIS packages. Within the BIDS development environment, you first create an SSIS project and then create and develop new packages.

The remainder of this lesson explains using both methods to develop SSIS packages.

Using the Import And Export Wizard

With SQL Server 2008, you use the Import And Export Wizard to copy data without going through the process of creating an SSIS project. When you use the wizard, it generates an SSIS package that you can execute immediately or save and then manually modify or schedule.

You typically start the Import And Export Wizard through SSMS when you are connected to the SQL Server relational engine. SSMS is the SQL Server management tool designed primarily for managing databases, and you will be using SSMS many times in the lessons throughout this Training Kit. To launch SSMS, from the Start menu, select Microsoft SQL Server 2008 and then SQL Server Management Studio. Figure 1-1 shows the Connect To Server dialog box, where you first connect to the Database Engine.

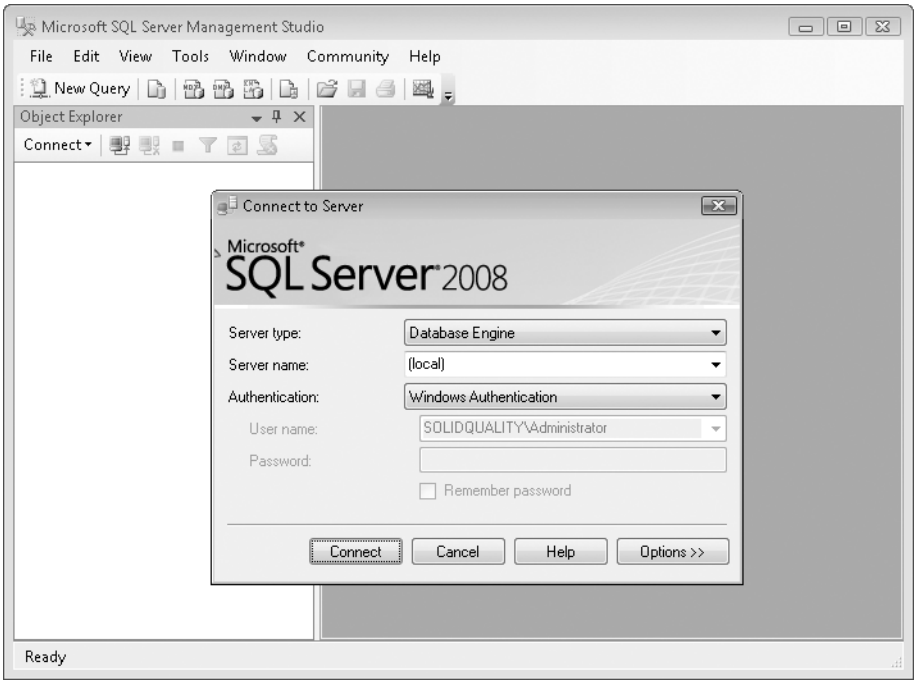


FIGURE 1-1 Start SSMS by selecting Microsoft SQL Server 2008 from the Start menu and then selecting SQL Server Management Studio. Connect to the Database Engine to manage SQL Server relational databases.

Starting the Import And Export Wizard

As its name states, the Import And Export Wizard can both import and export data. Use the following considerations to determine which part of the wizard to use:

- Importing data with the wizard lets you bring into a SQL Server table any data contained in accessible sources. Sources include other SQL Server databases, flat files, data in Microsoft Office Excel spreadsheets or Microsoft Office Access databases, and data in Oracle databases.

- Exporting data with the wizard lets you send data from SQL Server tables, views, or custom queries to flat files or database connections.

To start the Import And Export Wizard, follow these steps:

1. Through SSMS, connect to the instance of the SQL Server 2008 Database Engine that contains your source or your destination.
2. Open Object Explorer. You will find a list of various object containers under the SQL Server connection. The Databases folder shows all the databases attached to that instance of SQL Server. The System Databases subfolder contains the system databases.
3. To start the Import And Export Wizard, right-click the database that you want to use as your source or destination.
4. Click Tasks. If the database is the source of data that you want to send to a different system, select Export Data. If the database is the destination for files that currently exist outside the system, select Import Data, as Figure 1-2 shows.

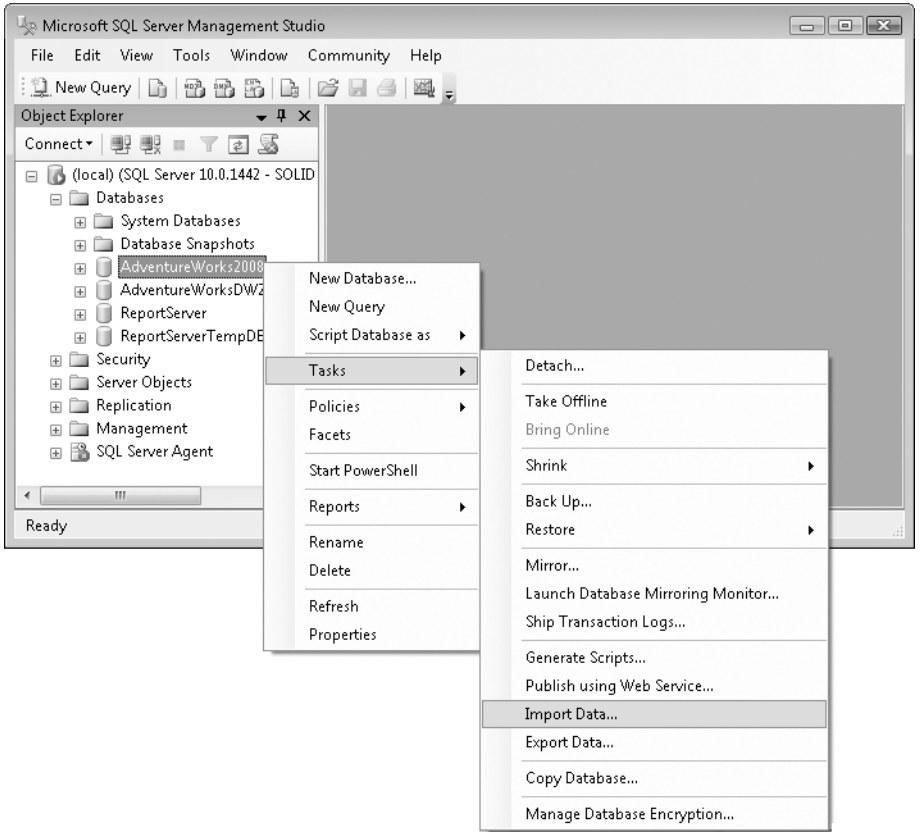


FIGURE 1-2 Start the Import And Export Wizard by right-clicking the database in SSMS and then clicking Tasks.

The wizard then walks you through several pages of questions, the answers to which are used to build the resulting package. The wizard pages include the following:

1. The Choose A Data Source page lets you specify where your data is coming from, such as a SQL Server database, an Excel file, a flat file, or other source. If your source is a relational database, you can also configure the security for the connection. Figure 1-3 shows the first page of the Import And Export Wizard.

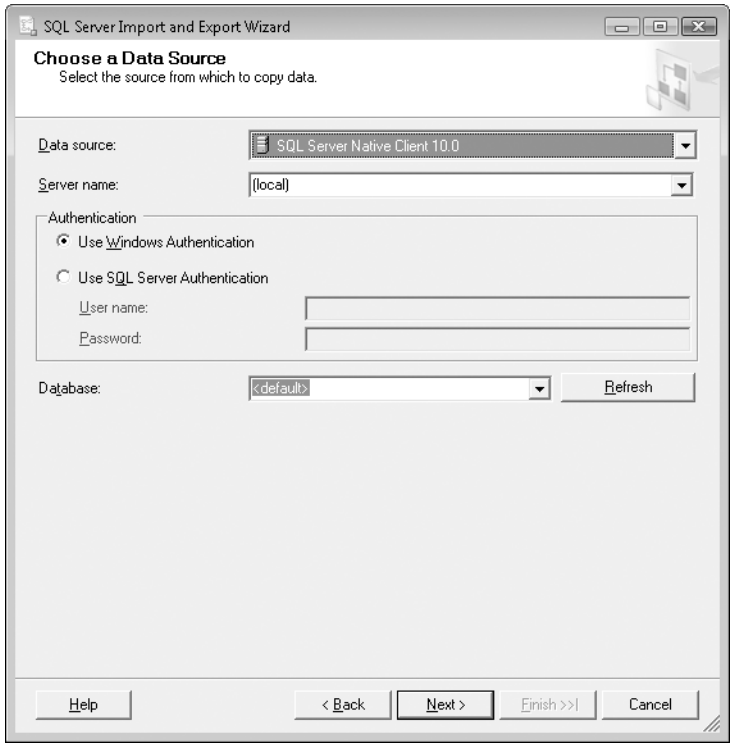


FIGURE 1-3 The Import And Export Wizard first lets you choose the data source where the data will be coming from, such as a SQL Server database, an Excel spreadsheet, or a flat file.

2. The Choose A Destination page lets you specify where your data will be sent. You specify the destination type and, if applicable, the server name and security settings needed to access the data. If you chose Import Data in SSMS to start the wizard, the data destination settings will match those of the database you selected prior to starting the wizard.
3. If you selected a relational database source that allows custom queries, on the Specify Table Copy Or Query page, you can choose to copy the data directly from the source to the destination or to specify a query. If you choose to specify a query, an additional page, named Provide A Source Query, enables you to manually enter the query.

4. If your source is a relational database and you do not specify a query, you can choose tables and views from your source on the Select Source Tables And Views page. If your source is a flat file or you specified a query, only the file or query is available as a choice. Also on this page, you can rename the destination table and edit the column mappings by clicking the Edit Mappings button to define column NULL settings, identity insert, and whether the table should be dropped and recreated every time.
5. Use the Save And Run Package page to execute the package immediately or save the package for later execution. If you save the package, you can later go back and edit the package by using the SSIS Designer, which is demonstrated in the rest of this chapter.

Saving and Editing Packages Created in the Wizard

The wizard's last page lets you execute the package immediately or save it. If you choose to save the autogenerated package within an Integration Services project in BIDS, as Figure 1-4 shows, you can modify its contents later. At times, you might want to use the wizard to generate a basic package to which you can add more advanced logic that the wizard cannot generate.

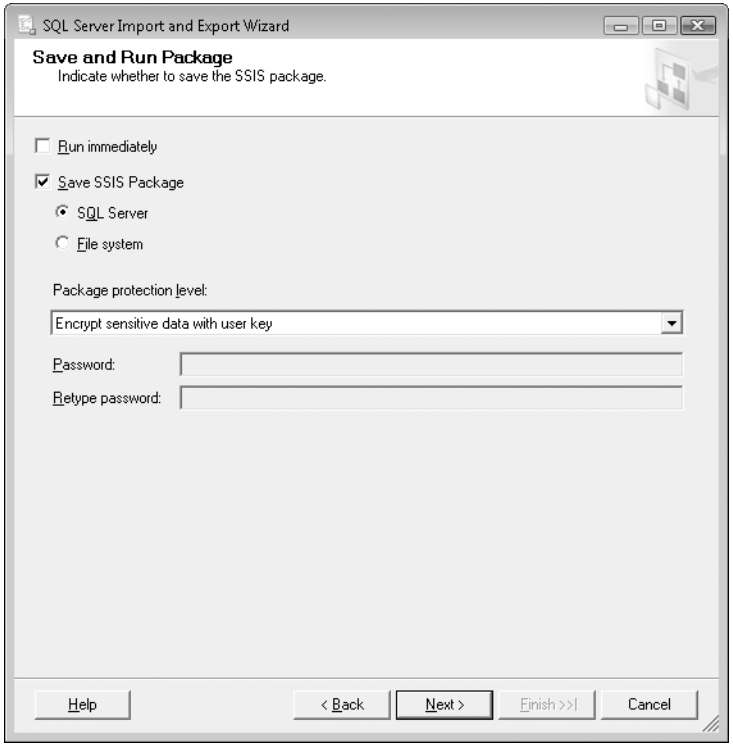


FIGURE 1-4 The final page of the Import And Export Wizard lets you execute and/or save packages.

In general, the Import And Export Wizard provides a quick way to move data from one source to a destination, especially for a one-time use, but there are some limitations:

- You can specify only one source and one destination in the wizard.
- Advanced workflow precedence is not available through the wizard.
- The wizard does not share data sources with other packages.

You need to evaluate whether your data processing requirements enable you to use the wizard or whether you need to develop a new package from scratch in BIDS.

Creating an SSIS Project in BIDS

Although the Import And Export Wizard is useful for generating a quick package that moves data from one source to one destination, these packages are frequently only a starting point. More often than not, you will need to either develop a package that has more complicated requirements or create a set of coordinated packages. For these cases, you first need to create a new SSIS project in BIDS.

> ***NOTE*** **OBJECTS IN BIDS**
>
> **Remember that any one project in BIDS can contain only objects from the same project type, such as SSIS, SQL Server Analysis Services (SSAS), or SQL Server Reporting Services (SSRS). However, a single project can be associated with projects of different types in the same solution.**

All of the SQL Server BI components are generated in a similar fashion through the BIDS development tool. To launch BIDS, from the Start menu, select Microsoft SQL Server 2008 and then SQL Server Business Intelligence Development Studio. Follow these steps to create a new SSIS project:

1. In BIDS, choose New, Project from the File menu. (If you have Visual Studio 2008 installed separately from BIDS, you can simply select New Project from the File menu.) Figure 1-5 shows the resulting New Project dialog box.
2. Fill out the New Project dialog box as follows:
   a. Under Project Types, select Business Intelligence Projects.
   b. Under Templates, select Integration Services Project.
   c. Assign a name to your project in the Name box.
   d. In the Location box, either leave the default folder location for storing new projects (in the ..\Documents\Visual Studio 2008\Projects\ folder) or change to a location of your choice.
3. When you have finished, click OK to build the project. The project contains several SSIS logical object folders, which Solution Explorer displays. You use these objects in your SSIS projects to point to connections and process data. Figure 1-6 shows a new project, with the default Package.dtsx package (created with the project) in the SSIS Designer and Solution Explorer on the right.

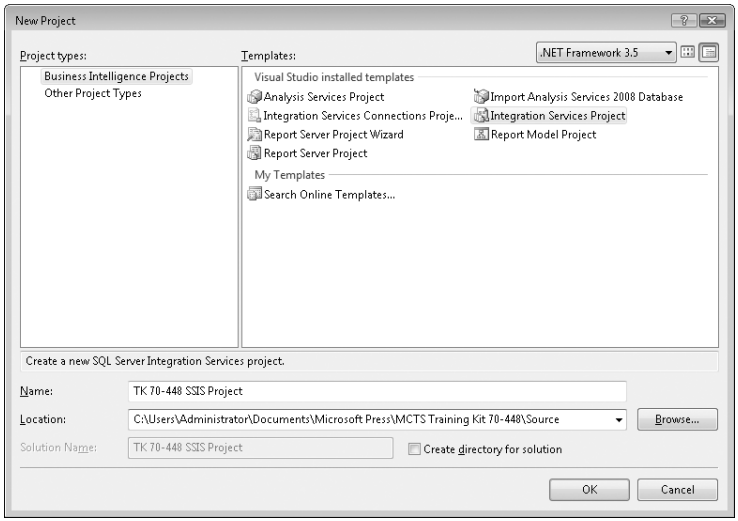


FIGURE 1-5 Creating a new project in BIDS begins in the New Project dialog box.

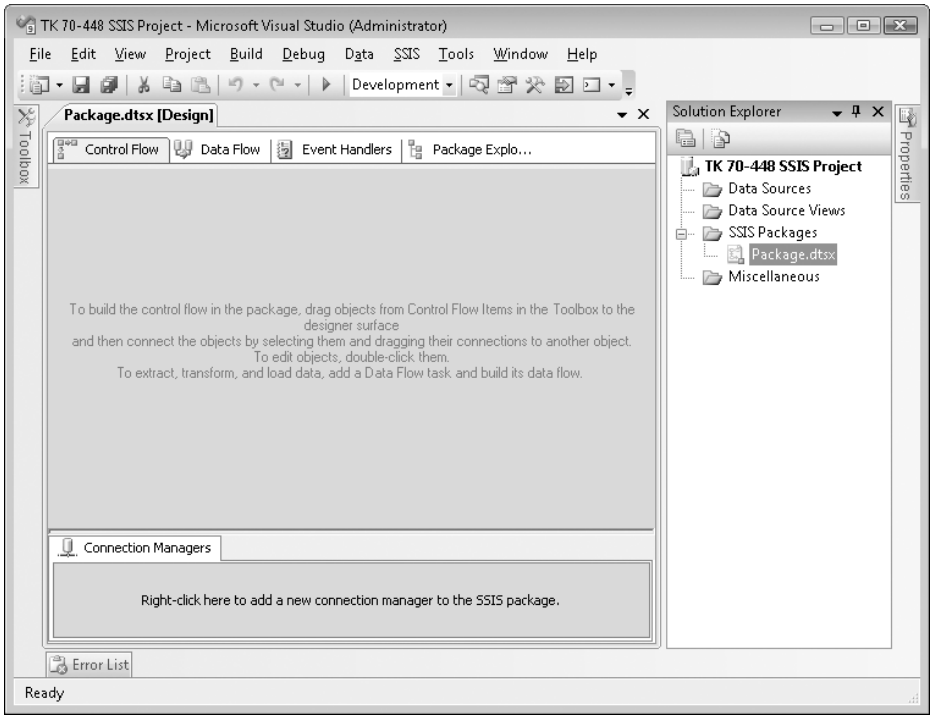


FIGURE 1-6 Creating a new project automatically creates a new SSIS package named Package.dtsx and several logical folders in Solution Explorer in BIDS.

You are now ready to configure and develop your package.

To add an existing package—such as one created by the Import And Export Wizard—to your project, right-click the SSIS Packages folder in Solution Explorer, and then click Add Existing Package. This dialog box lets you import packages from other projects or import packages that have already been deployed.

EXAM TIP

When you create packages in BIDS, the package is stored in the file system with the .dtsx file extension. This .dtsx file is an XML file that contains the logic and the layout for the design you have developed in BIDS, and you can move the file to a different project, manually deploy it to different servers, or make it part of a deployment package. In Chapter 3, "Deploying and Configuring SSIS Packages," and Chapter 4, "Administering, Securing, and Executing SSIS Packages," you will work with .dtsx files during deployment and execution.

Developing Project Data Sources and Package Connections

Because the main purpose of SSIS is to move data from sources to destinations, the next most important step is to add the pointers to these sources and destinations. These pointers are called *data sources* and *connections*. Data sources are stored at the project level and are found in Solution Explorer under the logical folder named Data Sources. Connections, on the other hand, are defined within packages and are found in the Connection Managers pane at the bottom of the Control Flow or Data Flow tab. Connections can be based on project data sources or can stand alone within packages. The next sections walk you through the uses and implementation of project data sources and package connections.

Creating a Data Source

A *data source* is an SSIS project object. Data sources contain connection strings that point to files or databases, and you can reference them within one or more packages. Data sources are optional within SSIS, but they are beneficial during development if you have a large number of packages that need to use the same database or file connection. Using a data source also helps if you need to change a connection used in many packages. You simply change the data source once and then open each package in your project, which will automatically synchronize the connection string stored in the package with the data source.

IMPORTANT **PROJECT DATA SOURCES ARE FOR DEVELOPMENT PURPOSES ONLY**

Be aware that after a package is deployed to a new environment and executed outside the project, the connection string is no longer updated by the project data source. Instead, you must use package configurations to share connection strings. See Chapter 3 to find out about sharing connection strings by using package configurations.

Using data sources in your project and packages is a two-step process:

1. **Creating the data source** Within Solution Explorer, right-click the Data Sources folder, and then click New Data Source. On the Welcome page of the wizard, click Next. Figure 1-7 shows the Data Source Wizard.

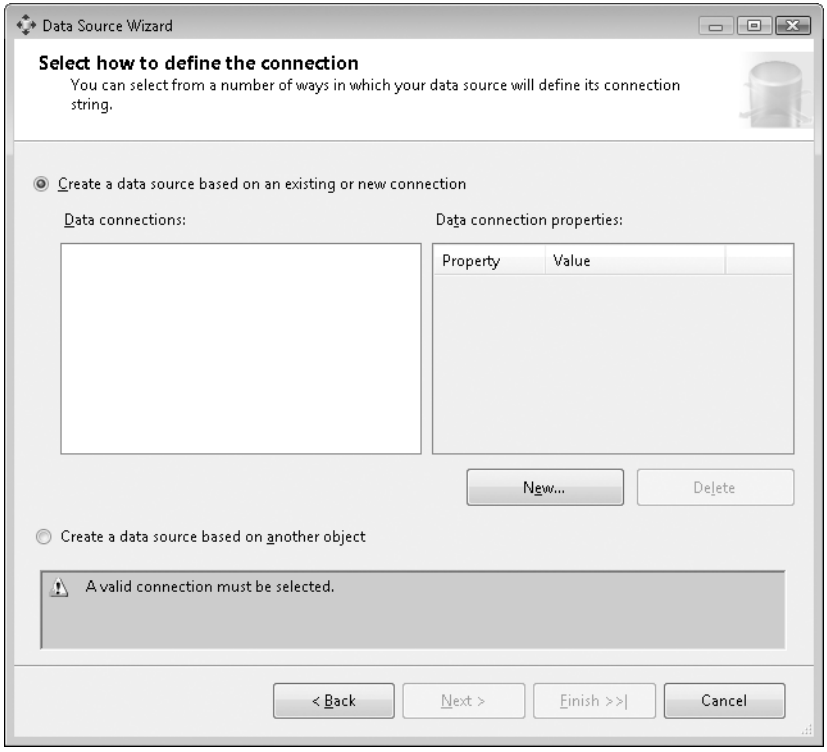


FIGURE 1-7 The Data Source Wizard lets you create a new connection in your project that can be shared between packages. The New button starts the Connection Wizard to create a new connection.

If you have made connections in the past on your server, a cached list of those connections appears in the Data Connections area, and you can choose an existing connection or click the New button to create a new connection. Figure 1-7 shows the connection page of the wizard without any cached connections.

2. **Adding the data source to a package** After you create your data source in the project, you need to add the data source to your packages.

Creating Package Connection Managers

A *package connection manager*, sometimes simply called a *package connection*, is independent of project data sources. However, package connections can reference a project data source. A package connection lets the different components in SSIS communicate with an object (such as a database, file, or server) outside the package. You can use package connections as source adapters, FTP or e-mail servers, or flat files.

If you link the package connection to the project data source, when the project data source is edited, the package connection is also updated when the package is being developed. In the BIDS design environment in Figure 1-8, Solution Explorer shows a project data source, and two package connections appear in the Connection Managers pane at the bottom of the SSIS Designer.

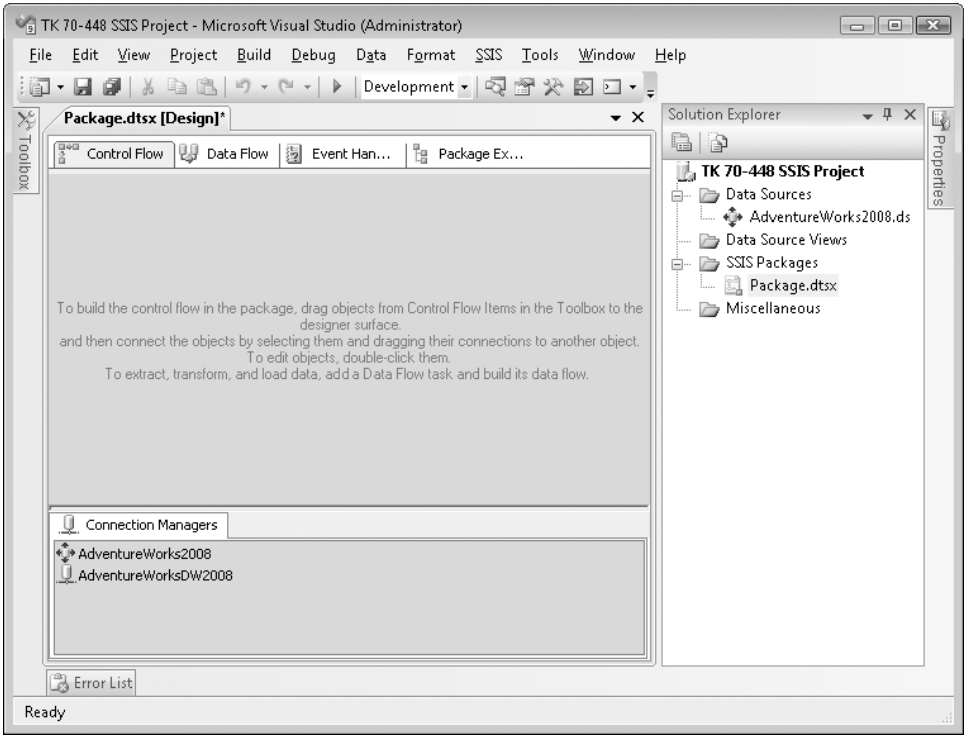


FIGURE 1-8 SSIS projects contain data sources, and packages contain connection managers. Project data sources can be linked to package connection managers, or connection managers can stand alone within a package.

Packages can be based on data sources defined in the SSIS project, or they can be stand-alone connections within a project. In Figure 1-8, the project has a data source named AdventureWorks2008, which is also referenced in the package's Connection Managers pane. In this example, the package contains another connection named AdventureWorksDW2008, which does not reference a project data source. The icon used to identify a project data source matches the icon for the package connection if a package connection references a project data source.

Adding Connections in the Connection Managers Pane

To create a new connection, right-click in the Connection Managers pane at the bottom of the Control Flow tab, as Figure 1-9 shows.

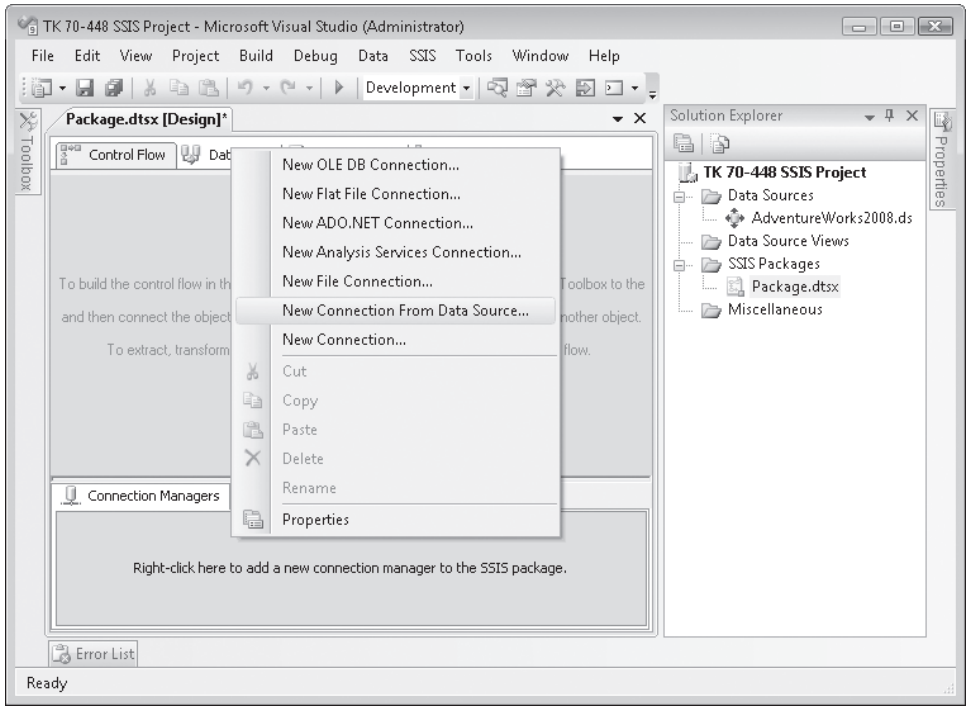


FIGURE 1-9 Right-click in the Connection Managers pane to create a new connection for your package.

You can create a connection from a project data source by selecting New Connection From Data Source, as Figure 1-9 shows, or you can create a stand-alone connection within a package by selecting New Connection and then choosing another connection provider from the Connection Manager Type list. A stand-alone connection is a connection that is not shared during development with other packages using a project data source. Stand-alone connections in a package that have the same name as a connection or connections in another package can, however, be updated together at run time through package configurations.

IMPORTANT **MODIFYING A PROJECT DATA SOURCE**

When you create a package connection from a data source, that connection is updated only during development whenever the package is opened and the data source has been changed. Package connections are not updated when they are run separately from the associated SSIS project—for example, when they are run from the command line.

The first step in creating a package connection is choosing the connection type, as Figure 1-9 shows. If you select a connection based on a data source, the connection has been created. However, if you choose another type of connection, you must perform at least one more step before the connection is complete. For example, if you are connecting to a relational database, you must select either New OLE DB Connection or New ADO.NET Connection

(depending on which connection provider you want to use to access an underlying database). After you select the connection type, you need to configure that connection.

When you select the connection type, SSIS prompts you to either select a connection string that has been cached on the machine you are working on or create a new cached connection. If the connection string is already cached on your machine, simply select that connection from the list to add it to the list of connections in the Connection Managers pane in your package.

If a connection string does not exist in the cache, you need to create a new connection string. For example, to define a new connection string for an OLE DB connection, in the Connection Managers pane, right-click and then click New OLE DB Connection. Next, in the Configure OLE DB Connection Manager dialog box, click New. In the Provider list, choose from the list of OLE DB providers that are installed on your machine, and then specify the database name and the connection security credentials, as Figure 1-10 shows, and then click OK. After you specify the connection options, you can choose the newly cached connection from the list, which then adds the new connection to the Connection Managers pane in the package.

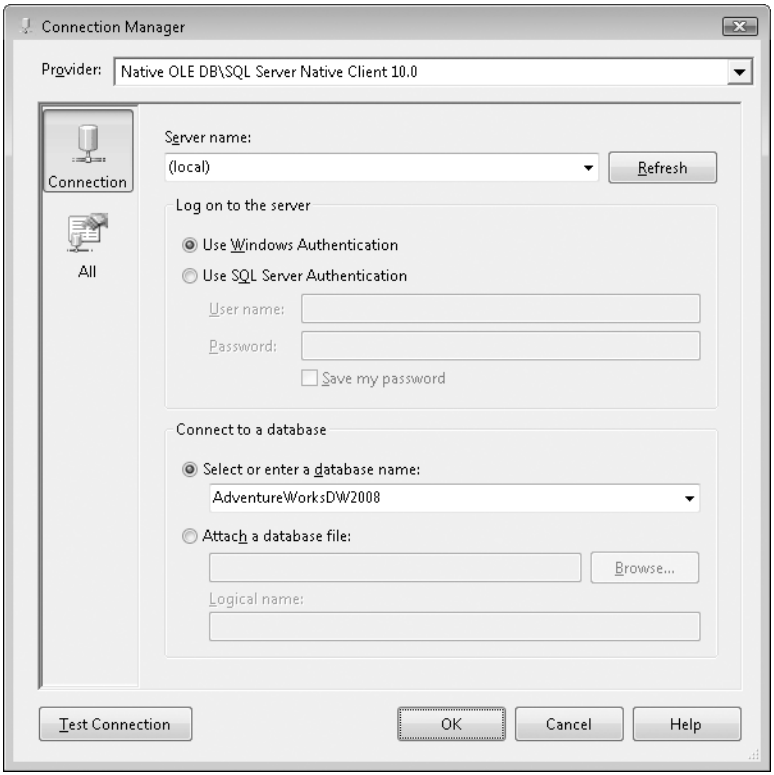


FIGURE 1-10 The Connection Manager dialog box lets you specify the provider, server, database, and security for the new connection.

Now that you have finished creating your package and package connections, you are ready to start developing package components. Connections are used in several package components, including control flow tasks, data flow adapters and transformations, event handlers, package logging, and package configurations—all of which are described in other chapters of this Training Kit.

PRACTICE Creating New Packages, Data Sources, and Connections

The following exercises familiarize you with the common tasks of creating a new project in BIDS and working with data sources and connections. All the practice files can be installed from the Practice folder on the companion CD.

EXERCISE 1 Create the Project and Packages

In this exercise, you will create a new SSIS project and then work with a couple of SSIS packages by adding data sources and connections.

1. Start SQL Server Business Intelligence Development Studio (BIDS), by clicking the Start button and then selecting All Programs, Microsoft SQL Server 2008, SQL Server Business Intelligence Development Studio.
2. Choose New, Project from the File menu. (If you have Visual Studio 2008 installed separately from BIDS, simply choose New Project from the File menu.) The New Project dialog box displays all the installed templates for Microsoft Visual Studio, including the Business Intelligence Projects templates.
3. In the New Project dialog box, confirm that Business Intelligence Projects is selected in the Project Types area, and then in the Templates area, select the Integration Services Project template.
4. Near the bottom of the New Project dialog box, in the Name box, type **TK 70-448 SSIS Project** as the name of your SSIS project.
5. In the Location box, type the path, starting with the Documents folder in your user profile: **..\Documents\Microsoft Press\MCTS Training Kit 70-448\Source**. This is the same location where the practice exercise files for the Training Kit will be installed by default.
6. Next, clear the Create Directory For Solution check box, which stores the SSIS project in the folder you specified in step 5.
7. Click OK to have BIDS create the new SSIS project.

8. When the project is created, SSIS automatically creates a new SSIS package named Package.dtsx and opens it in the SSIS Designer. In Solution Explorer, right-click Package.dtsx, and then click Rename.
9. Rename the package by typing **MyPackage.dtsx**. BIDS might prompt you to rename the package object. If a message box appears that prompts you to rename the package object as well, click Yes. Always click Yes if you are prompted to change the package object when renaming a package because this updates the internal name of the package.
10. Click the Save button on the toolbar, and then close the package by clicking the Close button in the upper-right corner of the SSIS Designer.
11. To create a new package, right-click the SSIS Packages folder in Solution Explorer, and then click New SSIS Package. This creates a new package object named Package1.dtsx (the number depends on how many packages you have created) in the SSIS Packages folder in Solution Explorer.
12. To rename the new package, right-click the package, and then click Rename. Rename the package to **DimCustomer.dtsx** because this package will contain logic to process the customer dimension table. When prompted, click Yes to rename the package object.
13. Following the same steps, create one more package in your SSIS Project named **DimPromotion.dtsx**.

EXERCISE 2 Create Project Data Sources

In this exercise, you will create two project data sources, which will be used in your packages as the source and the destination.

1. If necessary, start SQL Server Business Intelligence Development Studio (BIDS), open the project you created in Exercise 1, *TK 70-448 SSIS Project*, and then open Solution Explorer (if it is not already displayed). You can open Solution Explorer by clicking the Solution Explorer button on the Standard toolbar.
2. In Solution Explorer, right-click the Data Sources folder, and then click New Data Source. When the Welcome page of the Data Source Wizard appears, click Next.
3. On the Select How To Define The Connection page, select Create A Data Source Based On An Existing Or New Connection.
4. Click New to open the Connection Manager dialog box.
5. In the Provider drop-down list, select the Native OLE DB\SQL Server Native Client 10 provider and click OK. Type **(local)** in the Server Name field.
6. In the Select Or Enter A Database Name drop-down list, select AdventureWorks2008.

7. Click the Test Connection button, and then click OK. Click OK again to close the Connection Manager dialog box.
8. Select the (local).AdventureWorks2008 data connection in the Data Connections list, and then click Finish in the Data Source Wizard.
9. The Completing The Wizard page prompts you to enter a name for the new project data source. Type **AdventureWorks2008** in the Data Source Name box, and then click Finish. Be sure to remove the space between Adventure and Works2008.
10. Next, repeat steps 2 to 9 to create a new project data source for the (local).AdventureWorksDW2008 database, and name this data source **AdventureWorksDW2008.**
11. When you are finished creating the data sources, click the Save All button on the BIDS toolbar.

EXERCISE 3 Create New Package Connections from the Project Data Sources

In this exercise, you will add the project data sources you just created to the two packages that you have developed.

1. If necessary, start SQL Server Business Intelligence Development Studio (BIDS), open the project you created in Exercise 1, *TK 70-448 SSIS Project*, and then open Solution Explorer. Edit your MyPackage.dtsx package by double-clicking the package in Solution Explorer.
2. Locate the Connection Managers pane (at the bottom of the SSIS Designer window), right-click in the pane, and then click New Connection From Data Source.
3. In the Select Data Source dialog box, select both the AdventureWorks2008 and AdventureWorksDW2008 data sources from the list, and then click OK to accept. This puts the two project data sources into the package's Connection Managers pane.
4. Perform the same steps in the *DimCustomer.dtsx* package to add the AdventureWorks2008 and AdventureWorksDW2008 project data sources as connection managers for the package.
5. When you are finished creating the connection managers, click the Save All button on the BIDS toolbar.

Quick Check

1. You are asked to combine data from an Excel workbook and a database table and then push the results to a fixed-width flat file. Can you accomplish this task by using the Import And Export Wizard?
2. You need to create both SSIS packages to process your data and SSAS cubes to perform analysis. Can you create both objects in a single project?
3. What is the difference between a project data source and a package connection?
4. If a connection references a data source and the data source is changed, when will the connection be updated?

Quick Check Answers

1. No. The Import And Export Wizard lets you work with only a single source and a single destination. To combine data merging or data cleansing tasks, you need to either create a new package specifically for that purpose or modify a package previously created by the wizard.
2. No. You cannot create both SSIS and SSAS objects in one project because BIDS does not let you combine objects used for different platforms. You need to build two separate projects in BIDS: one for the SSIS packages and another for the SSAS cubes and dimensions.
3. Both project data sources and package connections are connection strings. However, a data source resides outside the package and can be used as the connection reference for more than one package. A package connection does not have to be associated with a data source.
4. Connections are updated by their associated data sources only when the package is opened for editing in BIDS.

Lesson 2: Creating and Editing Control Flow Objects

Estimated lesson time: 45 minutes

Now that you have created an SSIS project, packages, and package connections, it is time to start using SSIS features for data integration and for processing logic. This lesson and Lesson 3, "Using Data Flow Adapters and Transformations," focus on the two main components within SSIS: the control flow and the data flow. The *control flow* is the workflow engine and contains control flow tasks, containers, and precedence constraints, which manage when tasks and containers execute. The *data flow*, in contrast, is directly related to processing and transforming data from sources to destinations.

This lesson looks at defining control flow objects with the Control Flow design surface. When you have an SSIS project open within BIDS and are designing a package, the tabs across the top of the SSIS Designer let you choose an SSIS component to work with. The Control Flow design surface is the first tab and displays a workspace where you configure control flow objects. There are three primary types of control flow objects:

- **Control flow tasks** Workflow objects that perform operational-level jobs
- **Control flow containers** Provide a grouping mechanism for tasks and other containers
- **Constraints** Let you connect tasks and containers and define execution ordering and precedence

Creating Control Flow Tasks

A *control flow task* is an SSIS component that performs an operation such as sending an e-mail message, executing an SQL statement, or copying a file from an FTP server. When a control flow task is complete, it either succeeded or failed. You use the control flow to coordinate the execution of tasks in parallel or to set precedence constraints based on the tasks' completion status. See Chapter 2, "Debugging and Error Handling in SSIS," to learn more about precedence constraints.

To create a new control flow task in your package, drag the task from the toolbox to the Control Flow tab in the SSIS Designer. Figure 1-11 shows the Toolbox window when the Control Flow tab is clicked in the SSIS Designer.

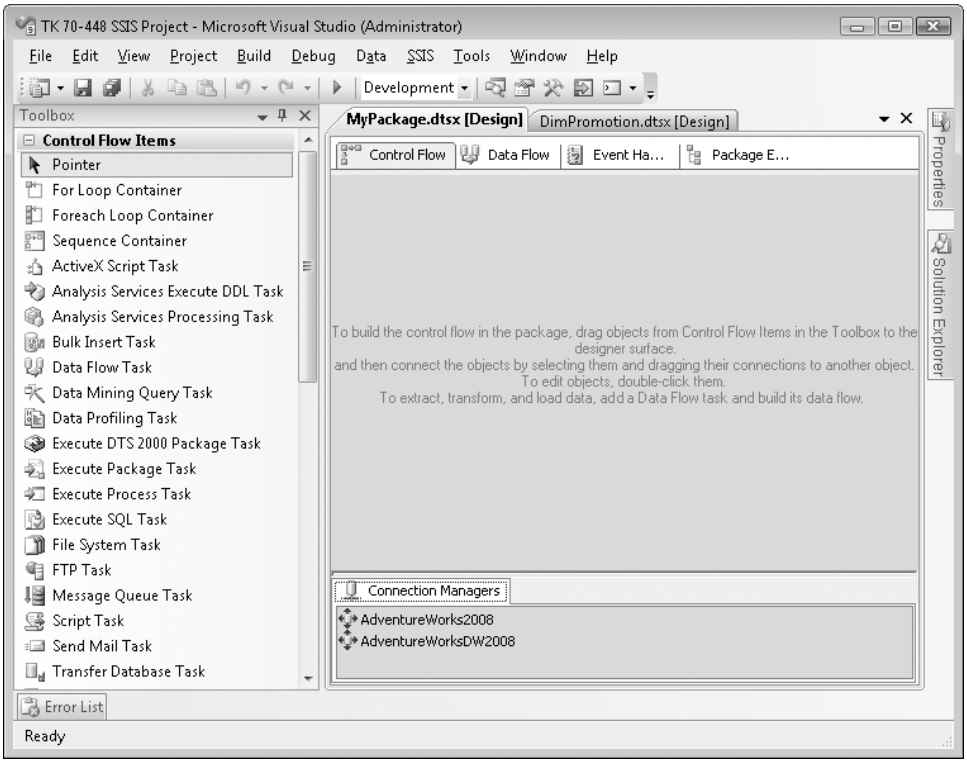


FIGURE 1-11 Control flow objects are found in the BIDS toolbox when you click the Control Flow tab in the SSIS Designer.

After you add a task to the control flow workspace, you need to configure the task to perform the specific operation you selected. To allow configuration, every task has an editor that you can open by double-clicking the task or by right-clicking the task and then clicking Edit. Table 1-1 lists the tasks in SSIS under the Control Flow Items list in the toolbox.

You might have noticed that there is also a list of Maintenance Plan Tasks for the control flow. These are primarily for database administrators (DBAs) who are managing SQL Server 2008 databases through the SSMS maintenance plan interface or DBAs who are creating packages in BIDS for database maintenance.

As you can see, SSIS features the ability to perform a host of different processing and integration operations. It is beyond the scope of this Training Kit and Exam 70-448 to discuss the design patterns for the components, but you will use several of these tasks in some lesson examples, and a couple of tasks are highlighted here.

TABLE 1-1 Common Tasks in SSIS

TASK	DESCRIPTION
ActiveX Script Task	Runs Microsoft Visual Basic Scripting Edition (VBScript) and JScript code and is included mainly for legacy support when a Data Transformation Services (DTS) package is migrated to SSIS.
Analysis Services Execute DDL Task	Runs XML for Analysis (XMLA) code against an SSAS database. XMLA is the data definition language (DDL) for SSAS; therefore, this task lets you perform common structure changes such as adding partitions to cubes.
Analysis Services Processing Task	Allows the processing of SSAS objects through an SSIS package.
Bulk Insert Task	Allows the execution of bulk copy operations for SQL Server. This task works only against a SQL Server Database Engine.
Data Flow Task	Allows data processing from sources to destinations. Lesson 3 in this chapter covers the data flow task in more detail.
Data Mining Query Task	Performs data mining queries and lets you capture the results for analysis.
Data Profiling Task	Allows the analysis of source data for patterns, missing data, candidate keys, and statistics. These results typically inform developers about what logic to include in their SSIS packages based on their data needs.
Execute DTS 2000 Package Task	Runs a DTS package within SSIS.
Execute Package Task	Runs other SSIS packages either deployed to SQL Server or in the file system.
Execute Process Task	Runs a command-line operation such as program or batch file execution.
Execute SQL Task	Runs SQL code against any underlying database connection in the SQL language of the connected database engine.
File System Task	Lets you copy, move, and delete files as well as perform other file and folder operations.
FTP Task	Sends and receives files between the file system and an FTP server and performs simple file and folder operations on the FTP server.

TASK	DESCRIPTION
Message Queue Task	Integrates with Message Queuing (MSMQ) on a server running Windows to read and send messages.
Script Task	Runs Microsoft Visual Basic 2008 or Microsoft Visual C# 2008 within an SSIS package.
Send Mail Task	Sends an e-mail message through an SMTP server.
Transfer *[Object]* Task	Tasks that copy SQL Server objects from one system to another, including databases, SQL Server Agent jobs, error messages, logins, master stored procedures, and database-level objects.
Web Service Task	Lets you connect to a Web service to send or receive information.
WMI Data Reader Task	Lets you run a Windows Management Instrumentation (WMI) query against the operating system to capture server information.
WMI Event Watcher Task	Waits for a particular event before executing.
XML Task	Combines, queries, and differentiates multiple XML files on the server.

Using Control Flow Containers

Your package must contain at least one task that performs a specific operation; however, most of the time, packages will have several tasks that coordinate with each other, and you need a way to organize those tasks. This is where a control flow container can help. A *control flow container* lets you group tasks together to control how tasks are run in parallel as well as ordering, logging, and transactions. Containers can also execute the tasks within them several times based on iterative requirements.

As with tasks, you find containers in the toolbox when you are working in the control flow. To use a container, you simply drag the container from the toolbox onto your control flow workspace. The screen in Figure 1-12 shows a package control flow containing a single container that holds several tasks.

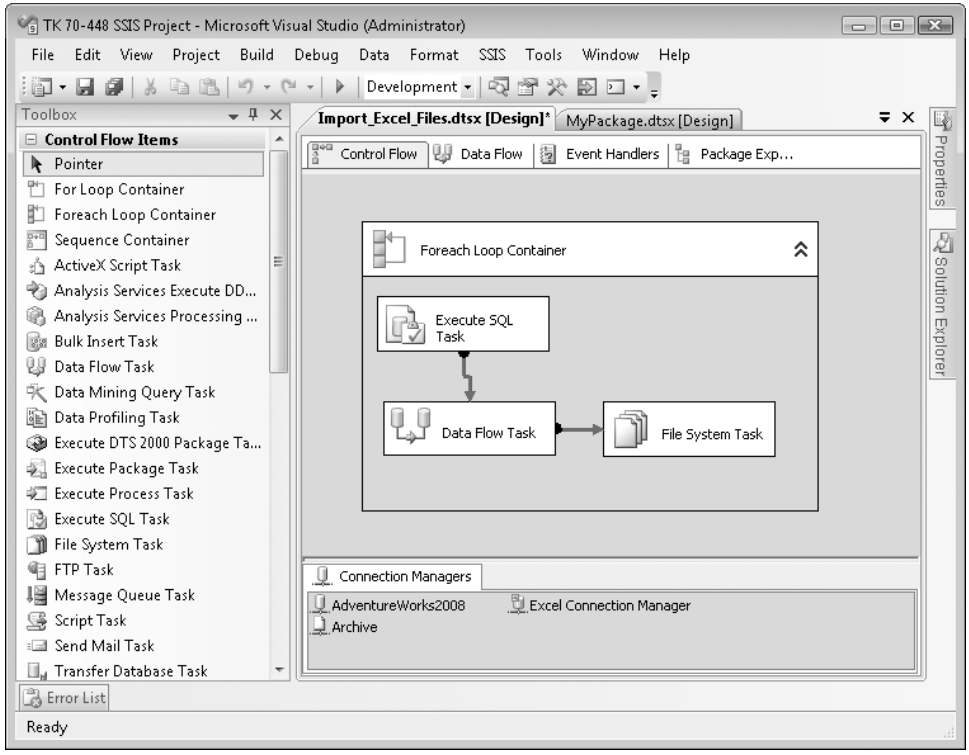


FIGURE 1-12 A control flow can include containers that group together tasks and subcontainers.

Additionally, you can drag task objects and other containers into your container.

There are three primary containers in SSIS: a Sequence Container, a For Loop Container, and a Foreach Loop Container.

- **Sequence Container** Lets you organize subordinate tasks by grouping them together, and lets you apply transactions or assign logging to the container.
- **For Loop Container** Provides the same functionality as the Sequence Container except that it also lets you run the tasks within it multiple times based on an evaluation condition, such as looping from 1 to 10.
- **Foreach Loop Container** Also allows looping, but instead of providing a condition expression, you loop over a set of objects, such as files in a folder.

Figure 1-13 shows the Foreach Loop Editor, which you open by double-clicking the container or by right-clicking the container and then clicking Edit.

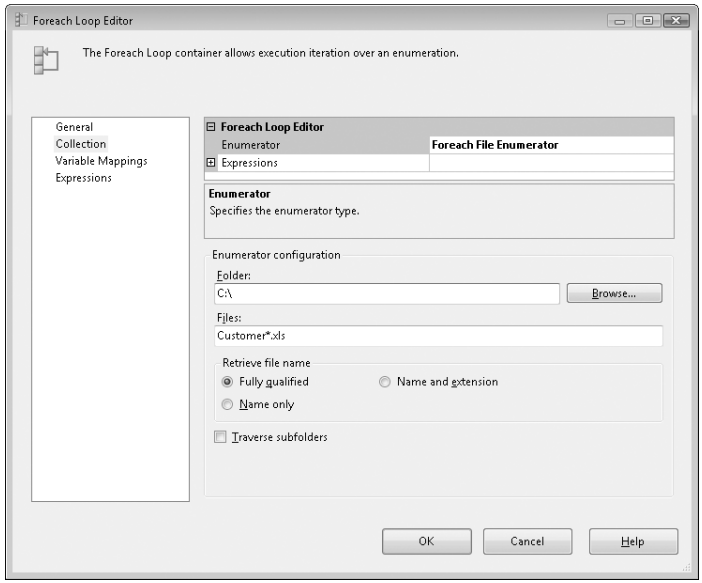


FIGURE 1-13 The Foreach Loop Editor lets you iterate over files in a folder and return the file names (one at a time) into a variable.

As described, the Foreach Loop Container can iterate over different types of objects, and the configuration choices let you specify the objects over which to loop and the detailed settings. Furthermore, the values of the enumerated objects can be put into user variables. For example, the Foreach Loop Container can iterate over files in a folder and return the file names into a variable.

Working with Package Variables

Variables within SSIS are a way to integrate objects by passing values between tasks and containers, accepting values from external sources, or building code dynamically that is then executed. You can also use variables for auditing and logging.

To work with variables within a package, choose Variables from the SSIS menu (when designing a package). Figure 1-14 shows the Variables window within BIDS.

At the top of the Variables window are buttons that let you create and delete variables as well as view other variables within a package. As Figure 1-14 shows, all SSIS variables are given a name, scope, data type, and value. The *scope* defines at what level within a package the variable is created. For example, if you select a Foreach Loop Container and then click the Add Variable button on the toolbar for the Variables window, the variable is scoped at that level. When no tasks or containers are selected, the variable is scoped at the entire package level.

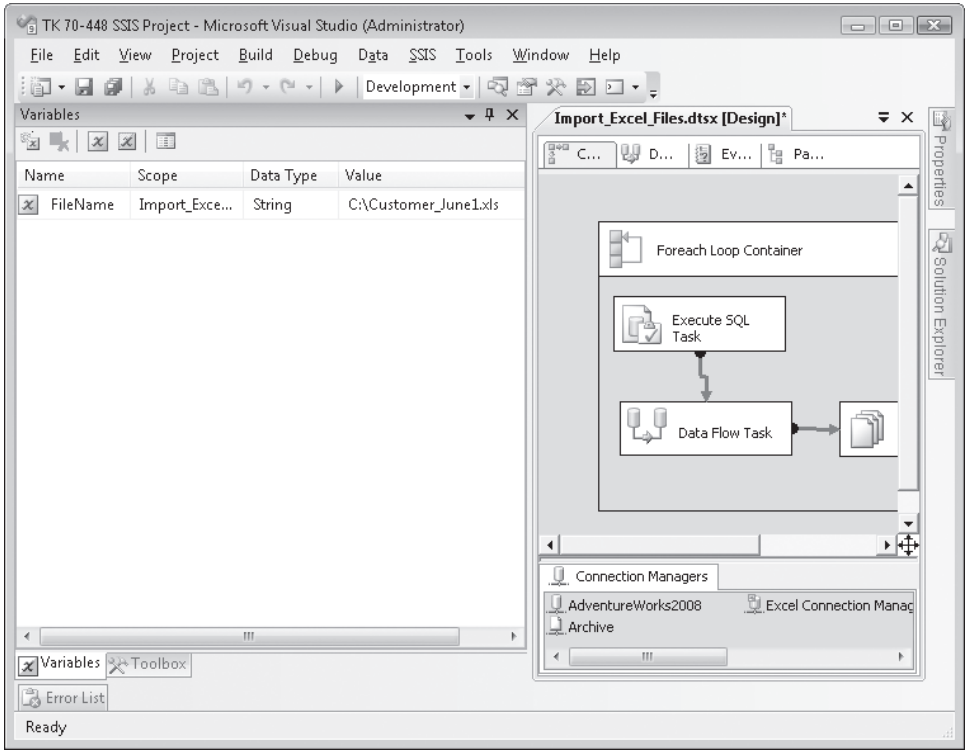


FIGURE 1-14 The Variables window lets you add, delete, or modify variables in a package.

EXAM TIP

Variables are viewable to a task or container only if the variables' scope is at the scope of the task or container in question, at the scope of a parent container level, or at the package scope itself. For example, if an Execute SQL Task has a variable directly assigned to it through the variable's scope, only the Execute SQL Task can see and use the variable. Other tasks or containers will not be able to reference the variable. On the other hand, if a Foreach Loop Container has a variable scoped to it, all the tasks within the Foreach Loop Container (including the container itself) can reference and use the variable. Variables are referenced as *User::[VariableName] or System::[VariableName]*.

Within SSIS, there are two types of variables: system variables and user variables.

- **System variables** System variables are not editable but can be referenced within tasks and containers. System variables are set by the package for tracking metadata such as the package name and the user that executes the package. To view all system variables, click Show System Variables (the button labeled with an X) on the Variables window toolbar.

- **User variables** You can create and define user variables for any purpose in the package. For example, the Foreach Loop Container updates user variables with values for every loop it iterates through. In Figure 1-13, shown earlier, file names for all files on the C drive that begin with the word *customer* will be mapped to a variable named *FileName*. Figure 1-15 shows the Variable Mapping tab in the Foreach Loop Editor.

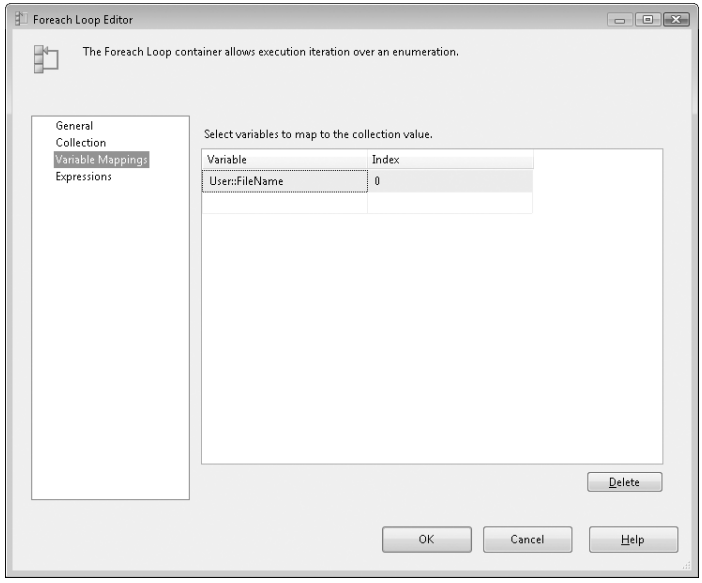


FIGURE 1-15 The Variable Mapping tab in the Foreach Loop Editor allows the values that are looped through to update an assigned variable for each iteration in the loop.

Using the Script Task and Data Profiling Task

Although this Training Kit focuses on development and maintenance rather than design, it is worth highlighting a few key control flow tasks. The exam objective domain covers using code within a package as well as performing data profiling, so let's look at the Script Task and the Data Profiling Task.

Script Task

You use the Script Task within SSIS to execute VB.NET or C#.NET code. The Script Task has the following features:

- Uses the Visual Studio Tools for Applications 2.0 (VSTA) interface, which lets you run VB.NET and C#.NET code with the full host of methods and functions.
- Variables can be referenced and updated within a script.
- Connections can be referenced and updated within a script.
- SSIS breakpoints can be applied within the script's code (for the Script Task). Chapter 2 will discuss breakpoints.
- Runs in both a 32-bit environment (X86) and a 64-bit environment (X64).

EXAM TIP

If you want to reference SSIS variables within a Script Task, you need to include the variables in the ReadOnlyVariables or ReadWriteVariables list, depending on whether you will be just accessing the variable for read purposes or updating the variable.

In the control flow example shown earlier in Figures 1-13, 1-14, and 1-15, the package is looping through Excel files and storing the Excel file path in a package variable. The package contains a connection to Excel that needs to be updated with the value of the variable, because each time the package loops, the variable needs to be updated.

Using a Script Task, you can update the Excel connection with the value of the variable. The first step is to drag a Script Task into the Foreach Loop Container from the toolbox. The script needs to be the first task that runs in the Foreach Loop Container, so place it before the Execute SQL Task and connect it to the Execute SQL Task with a precedence constraint. (Precedence constraints are covered in Chapter 2.) Figure 1-16 shows the Script Task Editor.

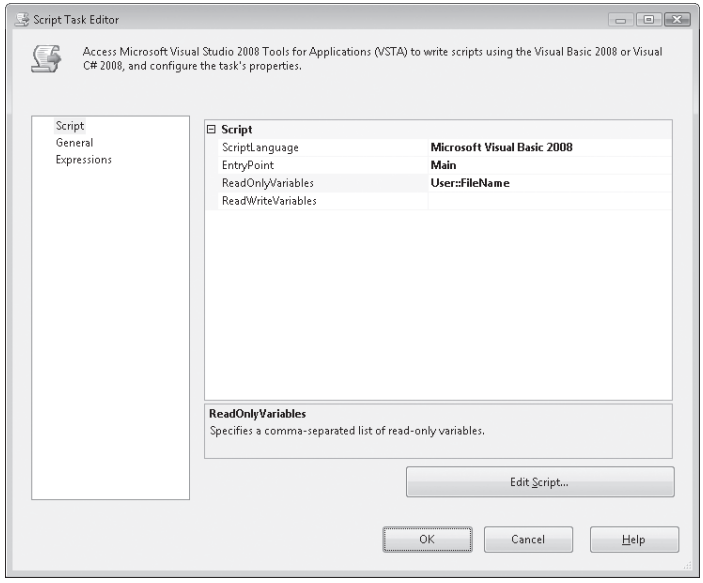


FIGURE 1-16 The Script Task Editor lets you select the programming language (VB.NET or C#.NET) as well as define any uses of variables within the script itself.

This example uses Microsoft Visual Basic 2008 as the *ScriptLanguage* and specifies the *User::FileName* variable in the *ReadOnlyVariables* property. To design the script, in the Script Task Editor, click Edit Script.

For this example, the script needs to update the Excel connection manager to point to the value of the variable, as the following code shows:

```
Dts.Connections("Excel Connection Manager").ConnectionString() = _
     "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + _
     Dts.Variables("FileName").Value.ToString() + _
     ";Extended Properties=""EXCEL 8.0;HDR=YES"";"
```

The reference to the connection begins with *Dts.Connections* and the reference to the variables begins with *Dts.Variables*.

This code executes for every loop in the Foreach Loop Container and updates the Excel connection manager.

Data Profiling Task

You use the Data Profiling Task to review source data entities, to check the cleanliness and completeness of the data, and to understand how the data is organized structurally, such as possible key columns and relationships between columns.

The Data Profiling Task has two parts: the Data Profiling Task in the control flow that performs the analysis and the Data Profile Viewer to review the results.

To use the Data Profiling Task, first create an ADO.NET connection where the source tables or views reside. The Data Profiling Task requires an ADO.NET connection for sources. Next, drag the task from the toolbox onto the control flow, and then open the task to edit its properties. The easiest way to perform a data profile is to click the Quick Profile button in the Data Profiling Task Editor. Figure 1-17 shows the Single Table Quick Profile Form dialog box configured to run against the [Sales].[vPersonDemographics] view.

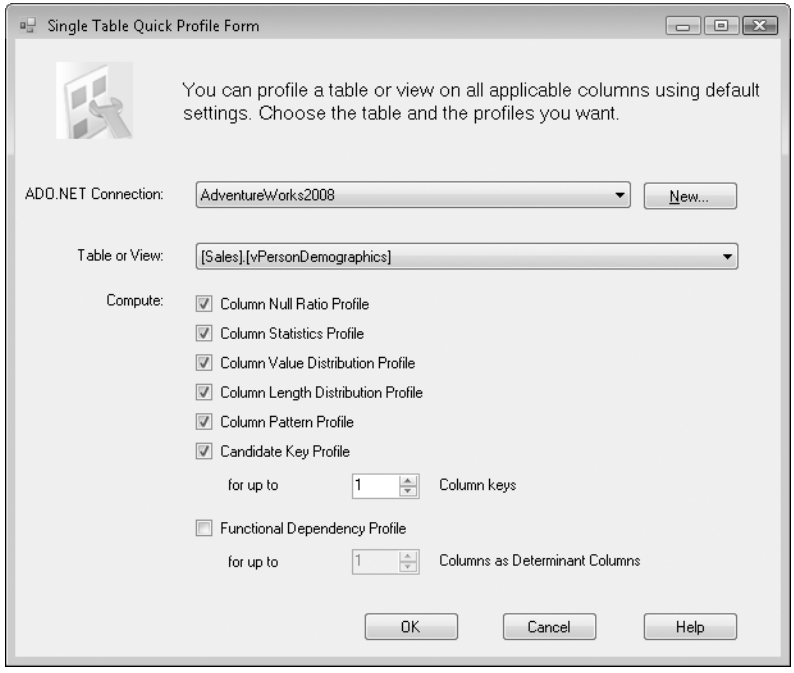


FIGURE 1-17 The Data Profiling Task can gather accuracy, completeness, and statistics information about the data within source tables or views.

As you can see, the Data Profiling Task can analyze data in various ways, which Table 1-2 describes.

TABLE 1-2 Data Profiling Task Features

PROFILE	DESCRIPTION
Column Null Ratio Profile	Evaluates the column and returns the percent of NULLs in the column relative to the total number of rows in the table.
Column Statistics Profile	For numeric and datetime columns, returns the spread and averages of the values.
Column Value Distribution Profile	Identifies the uniqueness of the values in a column across all the rows for that column.
Column Length Distribution Profile	Shows the various value lengths for a text column and the percentage of all the rows that each length takes up.
Column Pattern Profile	Displays any patterns found in the column data and returns the regular expression pattern that matches the pattern.
Candidate Key Profile	Identifies one or more columns that are unique across all the rows; the percentage of uniqueness is shown.
Functional Dependency Profile	Lists any columns that have value dependencies on other columns within the table, where a value from one column is found only when the value of another column is distinct.

After you configure the Data Profiling Task through the Single Table Quick Profile Form dialog box, you need to specify the output XML file in the *Destination* property. This is where the task stores the profiling results.

To view the results, open the Data Profile Viewer. (Click Start and then select All Programs, Microsoft SQL Server 2008, Integration Services, Data Profile Viewer.) Click Open on the toolbar, and browse to the output XML file where the results are stored. Figure 1-18 shows the Data Profile Viewer.

The Data Profile Viewer displays each profile type. The left pane lets you navigate between the profile types and source tables that were profiled. The right pane displays the results.

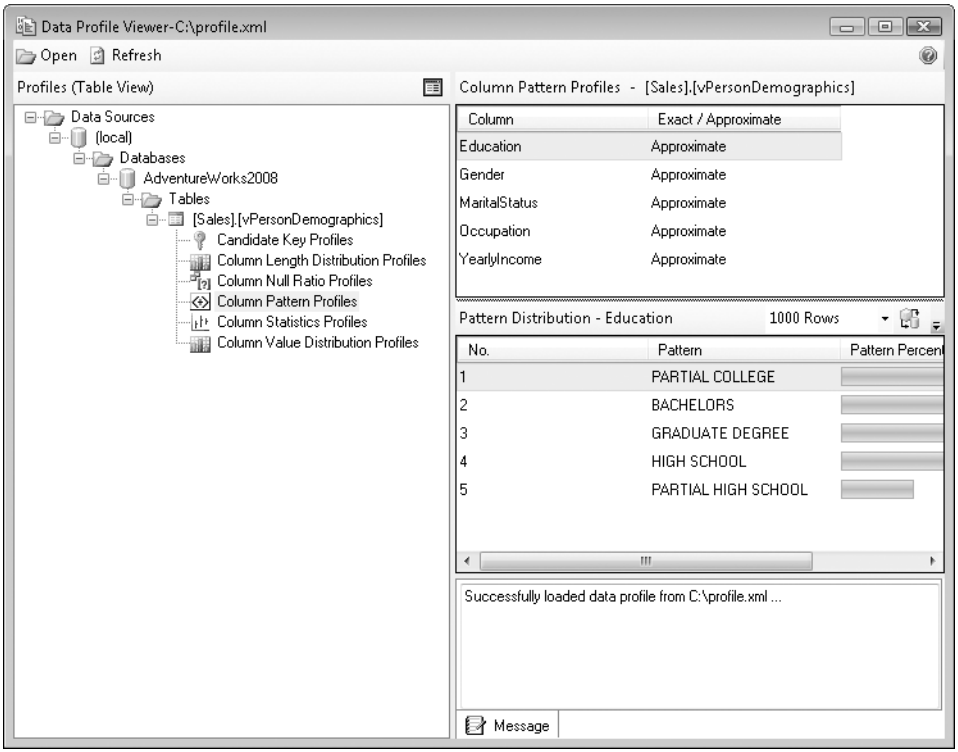


FIGURE 1-18 The Data Profile Viewer displays the results of the Data Profiling Task in a graphical form that demonstrates each profile type.

Testing Package Execution in BIDS

During the development of a package, you need to test its execution to validate that your package and tasks are configured correctly.

You can execute a package from within BIDS in three primary ways:

- Choose Start Debugging from the Debug menu on the menu bar.
- Click Start Debugging (the button containing an arrow that resembles the Play button on a DVD player) on the Standard toolbar.
- Press F5 on your keyboard.

After a package is run in BIDS, a new tab named the Progress tab appears in the SSIS Designer. This tab shows the execution results and lets you troubleshoot any package errors you might find. The Progress tab is renamed as Execution Results when you are back in design view.

PRACTICE Creating and Editing a Control Flow Task

The following exercises will familiarize you with creating and editing a control flow task and executing the package within the design environment.

EXERCISE 1 Create a Control Flow Task and Test Package Execution

In this exercise, you will work with control flow tasks and execute packages in the SSIS Designer.

1. If necessary, start SQL Server Business Intelligence Development Studio (BIDS), open the project *TK 70-448 SSIS Project* you created in Lesson 1, "Creating SSIS Packages and Data Sources," or open the completed exercise file from the companion CD, and then edit the package named MyPackage.dtsx (right-click the package in Solution Explorer, and then click Open).
2. Open the Toolbox window by selecting Toolbox from the View menu, locate the Execute SQL Task item, and drag it to the control flow workspace of your package.
3. Edit the Execute SQL Task object by double-clicking the task icon or by right-clicking the task icon and then clicking Edit.
4. Change the Connection property to use the AdventureWorks2008 connection.
5. In the SQL Statement property of the Execute SQL Task Editor dialog box, type the following code:

```
UPDATE Production.Product
SET ProductLine = 's'
WHERE ProductLine IS NULL
```

6. Click OK in the Execute SQL Task dialog box to return to the SSIS Designer. Right-click the Execute SQL Task, click Rename, and type **Update ProductLine**.
7. Next, drag a Sequence Container object from the toolbox onto the control flow workspace.
8. Drag the Update ProductLine Execute SQL Task you just created into the Sequence Container so that the task is nested in the Sequence Container box.
9. To test the execution of the package, click Start Debugging on the Standard toolbar or choose Start Debugging from the Debug menu.
10. When the package execution is complete, your Sequence Container and Execute SQL Task should be green.

11. Click the Execution Results tab (named Progress while the package is executing) in the SSIS Designer to view the execution details.
12. Select the Stop button from the tool menu to stop the debugger (or choose Debug, Stop Debugging from the Debug menu).
13. Click the Save All button on the BIDS toolbar.

EXERCISE 2 Modify the DimCustomer ETL Package Control Flow

In this exercise, you will start the process of building the DimCustomer SSIS package that will handle the ETL process from the AdventureWorks2008 database to the AdventureWorks-DW2008 database.

1. If necessary, start SQL Server Business Intelligence Development Studio (BIDS), open the project *TK 70-448 SSIS Project* you created in Lesson 1, "Creating SSIS Packages and Data Sources," or open the completed exercise file from the companion CD, and then open the empty DimCustomer package.
2. From the toolbox, drag two Execute SQL Tasks onto the control flow workspace and then drag one Data Flow Task onto the workspace.
3. Next, connect the first Execute SQL Task to the Data Flow Task by dragging the green precedence constraint from the Execute SQL Task onto the Data Flow Task. Then connect the green precedence constraint from the Data Flow Task to the second Execute SQL Task.
4. Rename the first Execute SQL Task to **Truncate Update Table**, and rename the second Execute SQL Task to **Batch Updates**. Figure 1-19 shows what your resulting control flow should look like.

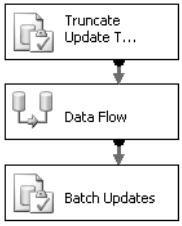


FIGURE 1-19 Your control flow for the DimCustomer package should contain an Execute SQL Task, followed by a Data Flow Task, followed by another Execute SQL Task.

5. Before editing the tasks in SSIS, open SSMS, connect to the Database Engine, and create a new query against the AdventureWorksDW2008 database. Execute the following code, which you can find in the CreateCustomerUpdateTable.sql file in the ..\Source\Ch 01\ folder of the practice exercise files.

```
USE AdventureWorksDW2008
GO

CREATE TABLE [dbo].[stgDimCustomerUpdates](
  [CustomerAlternateKey] [nvarchar](15) NULL,
  [AddressLine1] [nvarchar](60) NULL,
  [AddressLine2] [nvarchar](60) NULL,
  [BirthDate] [datetime] NULL,
  [CommuteDistance] [nvarchar](15) NULL,
  [DateFirstPurchase] [datetime] NULL,
  [EmailAddress] [nvarchar](50) NULL,
  [EnglishEducation] [nvarchar](40) NULL,
  [EnglishOccupation] [nvarchar](100) NULL,
  [FirstName] [nvarchar](50) NULL,
  [Gender] [nvarchar](1) NULL,
  [GeographyKey] [int] NULL,
  [HouseOwnerFlag] [nvarchar](1) NULL,
  [LastName] [nvarchar](50) NULL,
  [MaritalStatus] [nvarchar](1) NULL,
  [MiddleName] [nvarchar](50) NULL,
  [NumberCarsOwned] [tinyint] NULL,
  [NumberChildrenAtHome] [tinyint] NULL,
  [Phone] [nvarchar](25) NULL,
  [Suffix] [nvarchar](10) NULL,
  [Title] [nvarchar](8) NULL,
  [TotalChildren] [tinyint] NULL,
  [YearlyIncome] [nvarchar](100) NULL) ON [PRIMARY]
```

6. After you have successfully created the table, switch back to the DimCustomer SSIS package and edit the Execute SQL Task named Truncate Update Table.
7. In the Execute SQL Task Editor dialog box, set the Connection property to AdventureWorksDW2008, and then enter the following SQL code in the SQLStatement property before clicking OK to save it:

```
TRUNCATE TABLE dbo.stgDimCustomerUpdates
```

8. Edit the last Execute SQL Task, named Batch Updates, and set the Connection property to AdventureWorksDW2008.

9. In the SQLStatement property, enter the following UPDATE statement. (You can find this statement in the UpdateCustomerTable.sql file in the ..\Source\Ch 01\ practice exercise folder.)

```
UPDATE dbo.DimCustomer
   SET AddressLine1 = stgDimCustomerUpdates.AddressLine1
     , AddressLine2 = stgDimCustomerUpdates.AddressLine2
     , BirthDate = stgDimCustomerUpdates.BirthDate
     , CommuteDistance = stgDimCustomerUpdates.CommuteDistance
     , DateFirstPurchase = stgDimCustomerUpdates.DateFirstPurchase
     , EmailAddress = stgDimCustomerUpdates.EmailAddress
     , EnglishEducation = stgDimCustomerUpdates.EnglishEducation
     , EnglishOccupation = stgDimCustomerUpdates.EnglishOccupation
     , FirstName = stgDimCustomerUpdates.FirstName
     , Gender = stgDimCustomerUpdates.Gender
     , GeographyKey = stgDimCustomerUpdates.GeographyKey
     , HouseOwnerFlag = stgDimCustomerUpdates.HouseOwnerFlag
     , LastName = stgDimCustomerUpdates.LastName
     , MaritalStatus = stgDimCustomerUpdates.MaritalStatus
     , MiddleName = stgDimCustomerUpdates.MiddleName
     , NumberCarsOwned = stgDimCustomerUpdates.NumberCarsOwned
     , NumberChildrenAtHome = stgDimCustomerUpdates.NumberChildrenAtHome
     , Phone = stgDimCustomerUpdates.Phone
     , Suffix = stgDimCustomerUpdates.Suffix
     , Title = stgDimCustomerUpdates.Title
     , TotalChildren = stgDimCustomerUpdates.TotalChildren
  FROM dbo.DimCustomer DimCustomer
 INNER JOIN dbo.stgDimCustomerUpdates
    ON DimCustomer.CustomerAlternateKey
       = stgDimCustomerUpdates.CustomerAlternateKey
```

10. Click OK in the Execute SQL Task Editor dialog box, and then save the package. In the next lesson, you will complete the data flow portion of this package and then test the execution.

Quick Check

1. What is the difference between a control flow task and a control flow container?
2. To run a stored procedure within a SQL Server database, what task would you choose?

Quick Check Answers

1. Control flow tasks perform operations, whereas containers coordinate and group tasks. For example, a Foreach Loop Container can look through the files in a system folder, and a File System Task embedded within the container can then move the files to a new folder location.
2. The Execute SQL Task can run a stored procedure within SQL Server or any relational database for which you have an installed data provider. The syntax of the statement entered in the Execute SQL Task will be in the native language of the underlying database.

Lesson 3: Using Data Flow Adapters and Transformations

Estimated lesson time: 45 minutes

Lesson 2, "Creating and Editing Control Flow Objects," showed how to use control flow tasks and containers. One of the most valuable control flow tasks is the Data Flow Task. A package can have zero, one, or more data flows. To work with the Data Flow Task, you can either drag a Data Flow Task from the Control Flow toolbox onto the workspace and then double-click it, or you can click the Data Flow tab within the SSIS Designer. After clicking the Data Flow tab, you see the Data Flow Designer, where you can use the data flow to handle and transform datasets. The Data Flow Task has three types of objects in the toolbox:

- Data flow source adapters
- Data flow transformations
- Data flow destination adapters

Figure 1-20 shows the Data Flow tab with the toolbox open, highlighting the data flow sources and some of the data flow transformations. Notice the difference between the Control Flow toolbox items and the Data Flow toolbox items.

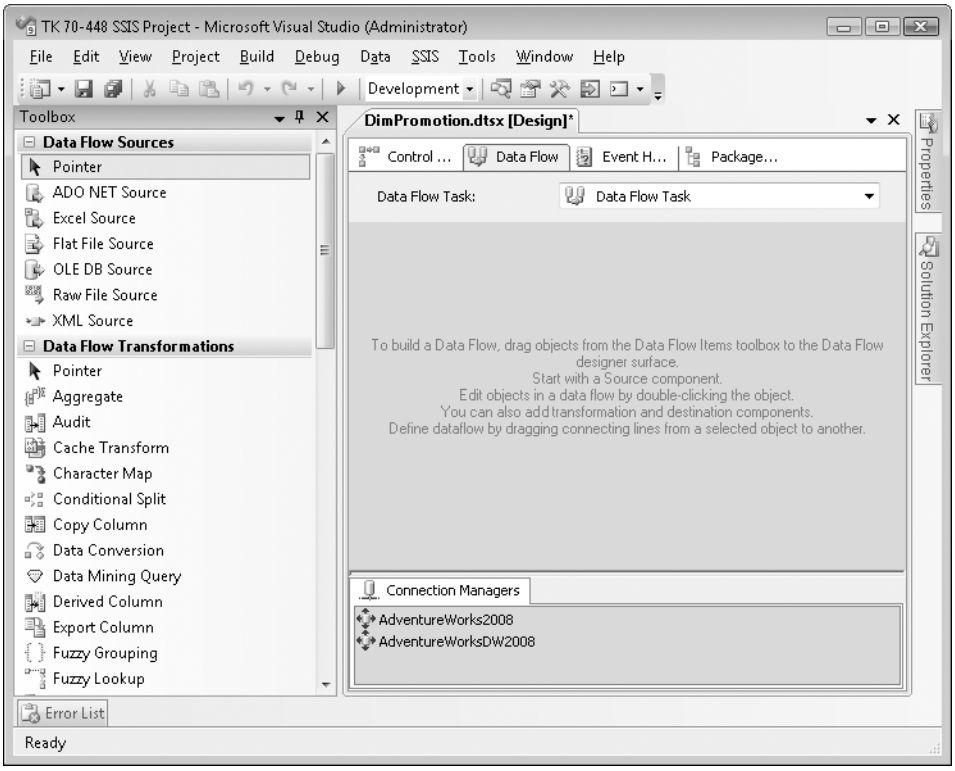


FIGURE 1-20 When you are working in the data flow, the toolbox shows items related to data flow development, including data flow sources, data flow transformations, and data flow destinations.

In this lesson, you will look at the details of the source and destination adapters as well as the transformations.

Defining Data Flow Source Adapters

Data flow source adapters use package connections, which point to the server instance or file location of the data source. (The only exception is the raw file adapter, which does not use a package connection.) A source adapter extracts data from sources and moves it into the data flow, where it will be modified and sent to a destination. You create data flow source adapters by dragging a source adapter from the Data Flow toolbox onto the Data Flow tab in the SSIS Designer. Table 1-3 describes the different data flow sources and their uses.

TABLE 1-3 Data Flow Sources and Their Uses

DATA FLOW SOURCE	PURPOSE
ADO.NET Source	Provides connections to tables or queries through an ADO.NET provider.
Excel Source	Allows extractions from an Excel worksheet defined in an Excel file.
Flat File Source	Connects to a delimited or fixed-width file created with different code pages.
OLE DB Source	Connects to installed OLE DB providers, such as SQL Server, Access, SSAS, and Oracle.
Raw File Source	Stores native SSIS data in a binary file type useful for data staging.
XML Source	Allows raw data to be extracted from an XML file; requires an XML schema to define data associations.

As an example, Figure 1-21 shows the OLE DB Source Editor dialog box for a package that is pulling special offer sales data from the AdventureWorks2008 database with the intention of loading it into the DimPromotions table in AdventureWorksDW2008.

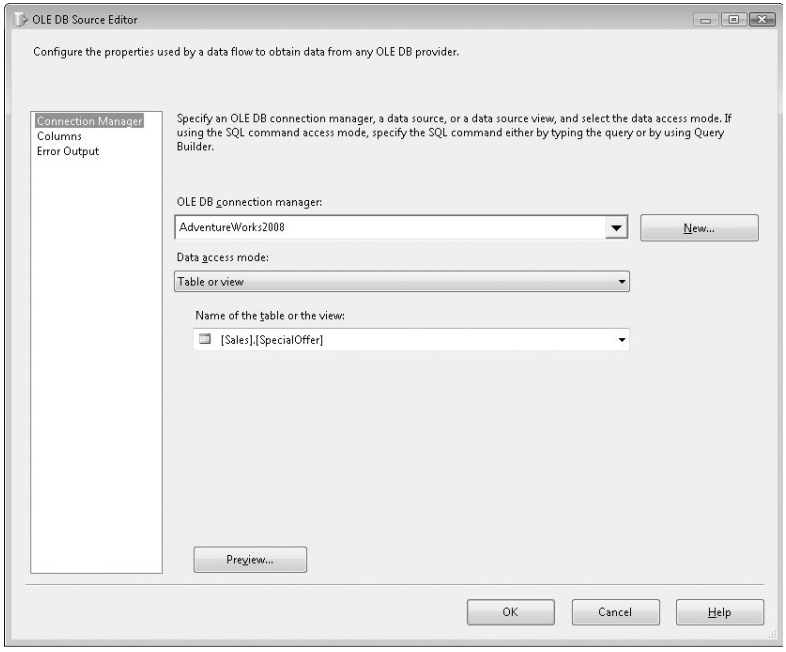


FIGURE 1-21 The OLE DB Source Editor displays that the DimPromotions package is configured to select data from the [Sales].[SpecialOffer] table.

The OLE DB source adapter is similar to the ADO.NET adapter in that a connection manager is required and either a table needs to be selected or a query needs to be written. In this example, the Data Access Mode drop-down list is set to Table Or View, and the [Sales].[SpecialOffer] table is selected. If the Data Access Mode was set to SQL Command, you could enter a query for the source.

Creating Data Flow Destinations

Data flow destinations are similar to sources in that they use package connections. However, destinations are the endpoints in a package, defining the location to which the data should be pushed. For example, if you are sending data to an Excel file from a database table, your destination will be an Excel Destination adapter.

EXAM TIP

Many SSIS objects have a *ValidateExternalMetadata* property that you can set to *False* if the object being referenced (such as a table) does not exist when the package is being designed. This property is most commonly used for source or destination adapters, such as when a destination table is created during package execution.

All the source adapters (except the Data Reader source) have matching destination adapters in the SSIS data flow. And there are other destination adapters that let you send data to even more destinations. Table 1-4 lists the destination adapters in the SSIS data flow.

TABLE 1-4 Data Flow Destinations and Their Uses

DATA FLOW DESTINATION	PURPOSE
ADO.NET Destination	Allows insertion of data by using an ADO.NET managed provider.
Data Mining Model Training	Lets you pass data from the data flow into a data mining model in SSAS.
DataReader Destination	Lets you put data in an ADO.NET recordset that can be programmatically referenced.
Dimension Processing	Lets SSAS dimensions be processed directly from data flowing through the data flow.
Excel Destination	Used for inserting data into Excel, including Excel 2007.
Flat File Destination	Allows insertion of data to a flat file such as a comma-delimited or tab-delimited file.
OLE DB Destination	Uses the OLE DB provider to insert rows into a destination system that allows an OLE DB connection.

DATA FLOW DESTINATION	PURPOSE
Partition Processing	Allows SSAS partitions to be processed directly from data flowing through the data flow.
Raw File Destination	Stores native SSIS data in a binary file type useful for data staging.
Recordset Destination	Takes the data flow data and creates a recordset in a package variable of type *object*.
SQL Server Compact Destination	Lets you send data to a mobile device running SQL Mobile.
SQL Server Destination	Provides a high-speed destination specific to SQL Server 2008 if the package is running on SQL Server.

EXAM TIP

You can configure the OLE DB Destination adapter to insert data from the data flow through bulk batches of data, instead of one row at a time. To use this destination-optimization technique, edit the OLE DB Destination and set the Data Access Mode to Table Or View—Fast Load. When the OLE DB Destination is not configured with fast load, only one row at a time will be inserted into the destination table.

Figure 1-22 shows a simple data flow with one source and one destination. The data flow extracts records from the AdventureWorks2008 SpecialOffers table and inserts them into the AdventureWorksDW2008 DimPromotions table.

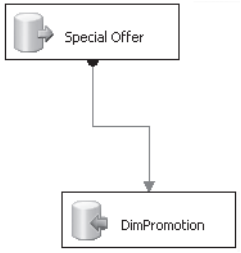


FIGURE 1-22 This simple data flow shows data being extracted from a source and inserted into a destination.

Like the source, the destination adapter requires configuration, both in the connection and table that the rows should be inserted into as well as in mapping the data flow columns to the destination table columns. Figure 1-23 shows the OLE DB Destination Editor for the preceding example.

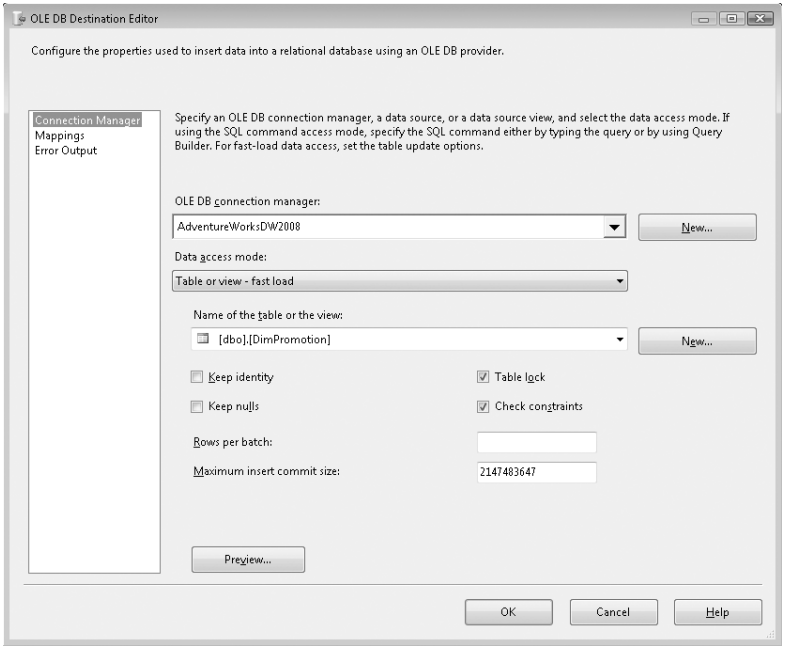


FIGURE 1-23 The destination adapter for the DimPromotions table is configured to connect to the AdventureWorksDW2008 database and insert rows into the DimPromotions table by using the fast-load feature.

Notice that this OLE DB Destination uses the AdventureWorksDW2008 connection and is configured by default to use the Table Or View—Fast Load option of the Data Access Mode drop-down list. This means that records will be processed with bulk insert statements rather than one row at a time.

Figure 1-24 shows the Mappings tab of the same OLE DB Destination Editor. This is where you map columns available from the data flow to the destination columns in the destination adapter. All the destination adapters have a Mappings tab.

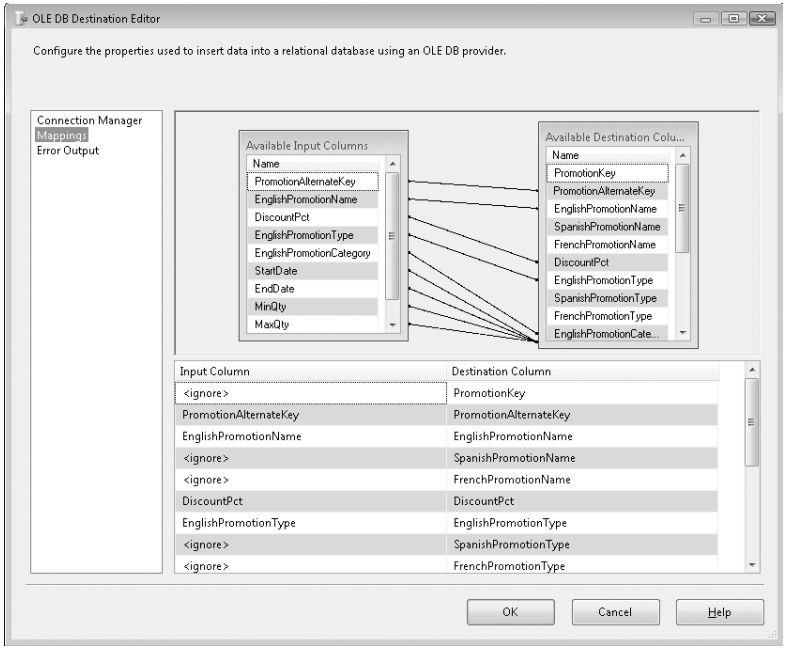


FIGURE 1-24 Each destination adapter requires you to map data from the data flow input columns to the destination columns.

Notice that not all columns are mapped. However, if one of the unmapped destination columns is marked as NOT NULL, the destination fails the package when it is run. In the section titled "Using Transformations" later in this lesson, you see how to use the Slowly Changing Dimension Transformation to handle new records and updates.

Working with Data Flow Transformations

Transformations give you the ability to modify and manipulate data in the data flow. A transformation performs an operation either on one row of data at a time or on several rows of data at once.

As with the source and destination adapters, you drag transformations from the Data Flow toolbox onto the Data Flow tab of the SSIS Designer, and edit them by right-clicking the transformation you want to change and then clicking Edit. You connect sources, transformations, and destinations through data paths, which you create by dragging the output arrow onto another component in the data flow. The green data path arrows are for rows that are successfully transformed, and the red output path arrows are for rows that failed the transformation because of an error, such as a truncation or conversion error. Figure 1-25 shows a data flow that connects a source to several transformations through data paths and onto a destination.

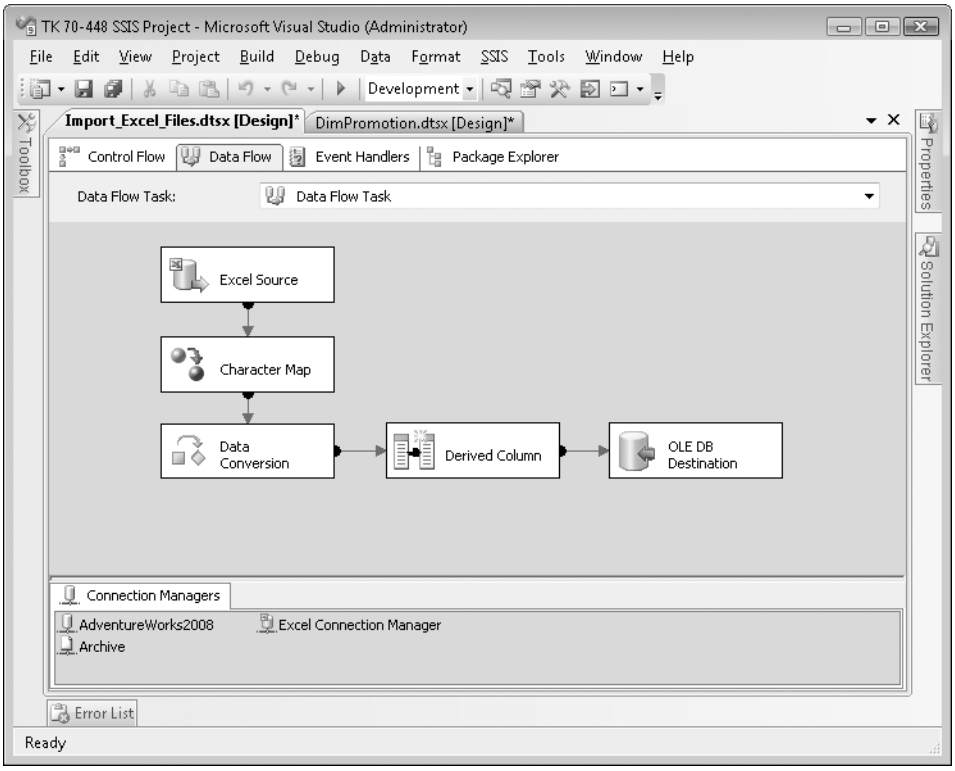


FIGURE 1-25 You use data paths to connect data flow transformations with sources, destinations, and other transformations.

Selecting Transformations

Transformations perform a wide variety of operations on the underlying data, and the transformation you choose depends on your data processing requirements. Some transformations operate similarly to other transformations; therefore, we can categorize them into natural groupings of like components.

LOGICAL ROW-LEVEL TRANSFORMATIONS

The most common and easily configured transformations perform operations on rows without needing other rows from the source. These transformations, which logically work at the row level, often perform very well. Table 1-5 lists the logical row-level transformations.

Some common uses of these transformations include performing mathematical calculations, converting a text value to a numeric or decimal data type, and replacing NULLs with other values. Because the Import Column and Export Column Transformations work with large binary data types, these two transformations carry an increased workload.

TABLE 1-5 Logical Row-Level Transformations

DATA FLOW TRANSFORMATION	PURPOSE
Audit	Adds additional columns to each row based on system package variables such as *ExecutionStartTime* and *PackageName*.
Cache Transform	Allows data that will be used in a Lookup Transformation to be cached and available for multiple Lookup components.
Character Map	Performs common text operations, such as Uppercase, and allows advanced linguistic bit conversion operations.
Copy Column	Duplicates column values in each row to new named columns.
Data Conversion	Creates new columns in each row based on new data types converted from other columns—for example, converting text to numeric.
Derived Column	Uses the SSIS Expression language to perform in-place calculations on existing values; alternatively, allows the addition of new columns based on expressions and calculations from other columns and variables.
Export Column	Exports binary large object (BLOB) columns, one row at a time, to a file.
Import Column	Loads binary files such as images into the pipeline; intended for a BLOB data type destination.
Row Count	Tracks the number of rows that flow through the transformation and stores the number in a package variable after the final row.

MULTI-INPUT OR MULTI-OUTPUT TRANSFORMATIONS

Multi-input and multi-output transformations can work with more than one data input or can generate more than one output, respectively. These transformations provide the capability to combine or branch data and give the data flow the overall ability to process data from one or more sources to one or more destinations. Table 1-6 lists the multi-input and multi-output transformations.

TABLE 1-6 Multi-Input and Multi-Output Transformations

DATA FLOW TRANSFORMATION	PURPOSE
Conditional Split	Routes or filters data based on Boolean expressions to one or more outputs, from which each row can be sent out only one output path.
Lookup	Allows matching between pipeline column values to external database tables; additional columns can be added to the data flow from the external table.
Merge	Combines the rows of two similar sorted inputs, one on top of the other, based on a defined sort key.
Merge Join	Joins the rows of two sorted inputs based on a defined join column(s), adding columns from each source.
Multicast	Generates one or more identical outputs, from which every row is sent out every output.
Union All	Combines one or more similar inputs, stacking rows one on top of another, based on matching columns.

As Table 1-6 describes, the Merge and Merge Join Transformations require sorted inputs. When these components are used in a data flow, the transformation waits for rows from either input, based on the defined sort order, to preserve the sorted output or match across the sorted rows; this means that rows might not immediately be sent out the output path.

EXAM TIP

When trying to determine which transformation to use that brings more than one data source together, remember that the Merge Join Transformation brings two sorted sources together and matching rows together with either an Inner Join, a Full Outer Join, or a Left Outer Join. Merge Join can match more than one row across the join columns. This behavior is different from that of the Lookup Transformation, which brings back only a single match across the join columns of the Lookup table.

The Union All Transformation does not join rows together but rather brings each row separately from the sources, stacking the rows together. The number of rows in the output of Union All is the combined row counts of all the inputs. The Merge Transformation is similar to Union All, except that the sources have to be sorted and the sort position is preserved.

MULTI-ROW TRANSFORMATIONS

Some transformations perform work based on criteria from multiple input rows or generate multiple output rows from a single input row. These transformations can be more intensive in operation and memory overhead, but they provide valuable functions to meet business requirements. Table 1-7 lists the multi-row transformations.

TABLE 1-7 Multi-Row Transformations

DATA FLOW TRANSFORMATION	PURPOSE
Aggregate	Associates records based on defined groupings and generates aggregations such as *SUM*, *MAX*, *MIN*, and *COUNT*.
Percent Sampling	Filters the input rows by allowing only a defined percent to be passed to the output path.
Pivot	Takes multiple input rows and pivots the rows to generate an output with more columns based on the original row values.
Row Sampling	Outputs a fixed number of rows, sampling the data from the entire input, no matter how much larger than the defined output the input is.
Sort	Orders the input based on defined sort columns and sort direction and allows the removal of duplicates across the sort columns.
Unpivot	Takes a single row and outputs multiple rows, moving column values to the new row based on defined columns.

In the cases of the Sort, Aggregate, and Row Sampling Transformations, all the input rows are blocked, allowing the transformations to perform the work before sending rows down the output path. These transformations often require more server resources, memory, and processor capacity than do other transformations.

ADVANCED DATA-PREPARATION TRANSFORMATIONS

The final grouping of transformations lets you perform advanced operations on rows in the data flow pipeline. Table 1-8 lists these advanced data-preparation transformations.

TABLE 1-8 Advanced Data-Preparation Transformations

DATA FLOW TRANSFORMATION	PURPOSE
OLE DB Command	Performs database operations such as updates and deletes, one row at a time, based on mapped parameters from input rows.
Slowly Changing Dimension	Processes dimension changes, including tracking dimension history and updating dimension values. The Slowly Changing Dimension Transformation handles these common dimension change types: Historical Attributes, Fixed Attributes, and Changing Attributes.
Data Mining Query	Applies input rows against a data mining model for prediction.
Fuzzy Grouping	Associates column values with a set of rows based on similarity, for data cleansing.
Fuzzy Lookup	Joins a data flow input to a reference table based on column similarity. The Similarity Threshold setting specifies the closeness of allowed matches—a high setting means that matching values are closer in similarity.
Script Component	Provides VB.NET scripting capabilities against rows, columns, inputs, and outputs in the data flow pipeline.
Term Extraction	Analyzes text input columns for English nouns and noun phrases.
Term Lookup	Analyzes text input columns against a user-defined set of words for association.

REAL WORLD

Erik Veerman

Most customers I've worked with have numerous systems that host all kinds of data—from sales transaction systems, to human resources, to custom business apps. Many of these systems even run on different database platforms such as SQL Server, Oracle, DB2, or legacy systems that aren't even sold any more. The complexity of data within organizations extends beyond just the enterprise relational database, often including Excel files and departmental Access applications. Navigating through these systems can be difficult, to say the least.

SSIS delivers real benefits in these situations because it helps you efficiently consolidate data.

In one customer engagement I was involved with, our task was to simplify a complicated process that pulled in five different data sources. Two of them were in SQL Server, one was in Oracle, another was in Excel, and the last was a large binary flat file created from an IBM AS/400 system.

Before we redesigned the processing of this data, the operation required a nightly job that ran for 7.5 hours. The job included a batch process to convert the AS/400 binary file to ASCII, and the job pulled the Oracle and Excel data into a staging environment inside SQL Server and then through a rather large stored procedure. Custom logic joined data (numbering in the millions of rows) across servers through linked servers and staged the data into about 15 staging tables before the finished product was produced. Sound familiar?

The redesign in SSIS reduced a lot of the complexities because we could extract data from these sources directly into the data flow in SSIS and join different sources together. We also were able to convert the complicated T-SQL logic involving the staging tables to a series of transformations and tremendously reduce the overall disk I/O by going from 15 to 3 staging tables.

The net result was three SSIS packages that ran together in 25 minutes. What a time gain. In addition, using SSIS reduced hardware overhead and management of the old process, allowing the customer's IT professionals to do much more with time they didn't think they could ever recoup.

Using Transformations

Each transformation has an editor window to define the way the operation is applied to the data. For example, the Derived Column Transformation specifies an expression that generates a new column in the data flow or replaces an existing column. To open the Transformation Editor, either double-click the transformation or right-click the transformation and then click Edit. Figure 1-26 shows the Derived Column Transformation Editor.

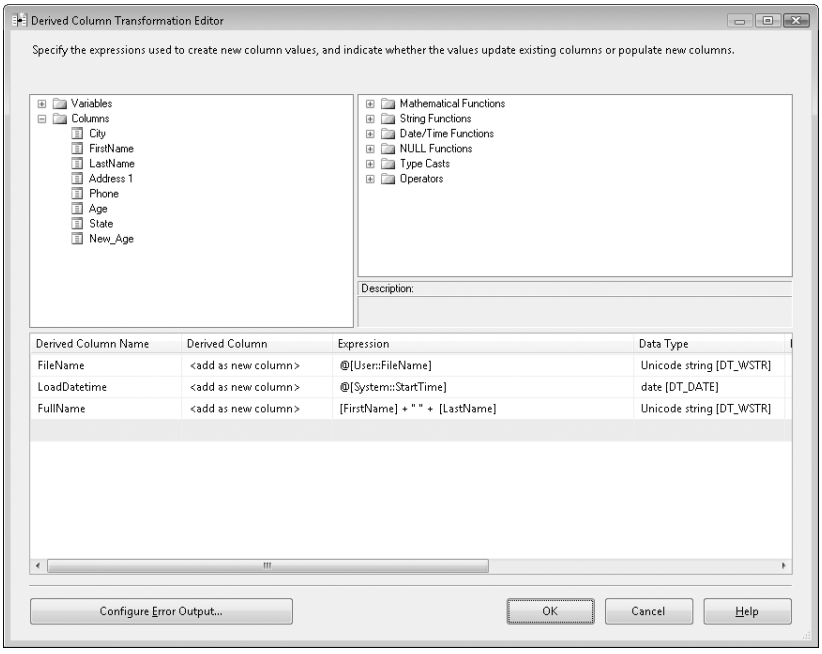


FIGURE 1-26 The Derived Column Transformation Editor specifies how the data is manipulated as it flows through a transformation.

In the Derived Column example in Figure 1-26, one of the new columns added to the data flow is named FullName, which is based on the concatenation of the FirstName column and the LastName column using the following SSIS expression:

```
[FirstName] + " " +  [LastName]
```

Other transformations contain similar functionality. Each transformation has an editor specific to the chosen operation.

The next example uses the Slowly Changing Dimension Transformation in the DimPromotion package to identify new records versus updated records. Figure 1-27 shows the data flow that results.

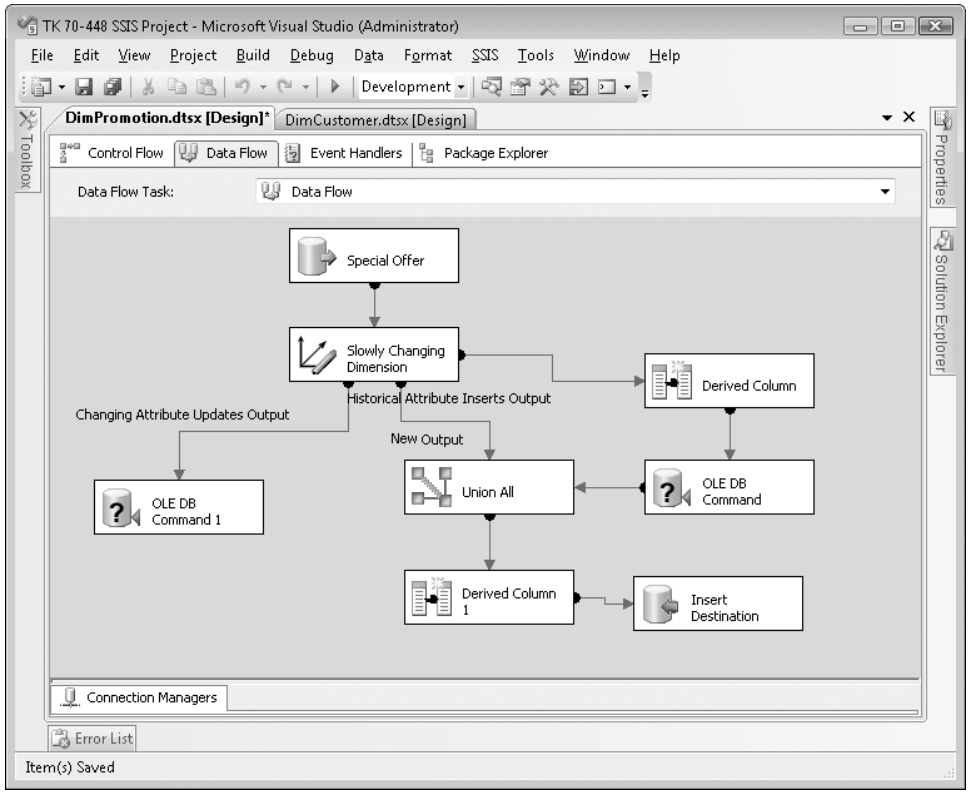


FIGURE 1-27 The Slowly Changing Dimension Transformation sends rows out multiple outputs depending on whether there is a new record or a change and on what kind of change.

Figure 1-27 shows the output of the Slowly Changing Dimension Transformation. All the output transformations and destinations were created by the Slowly Changing Dimension Wizard, which built the rest of the data flow.

Figure 1-28 shows the Slowly Changing Dimension Columns page of the wizard, which defines which dimension columns should cause what kind of change to the output. The options are Fixed Attribute, which means the change should not happen; Changing Attribute, which means that an update happens; or Historical Attribute, which means that the change creates a new record.

A detailed review of all the SSIS transformations is outside the scope of this Training Kit, but you can find information about them in the References section at the end of this book.

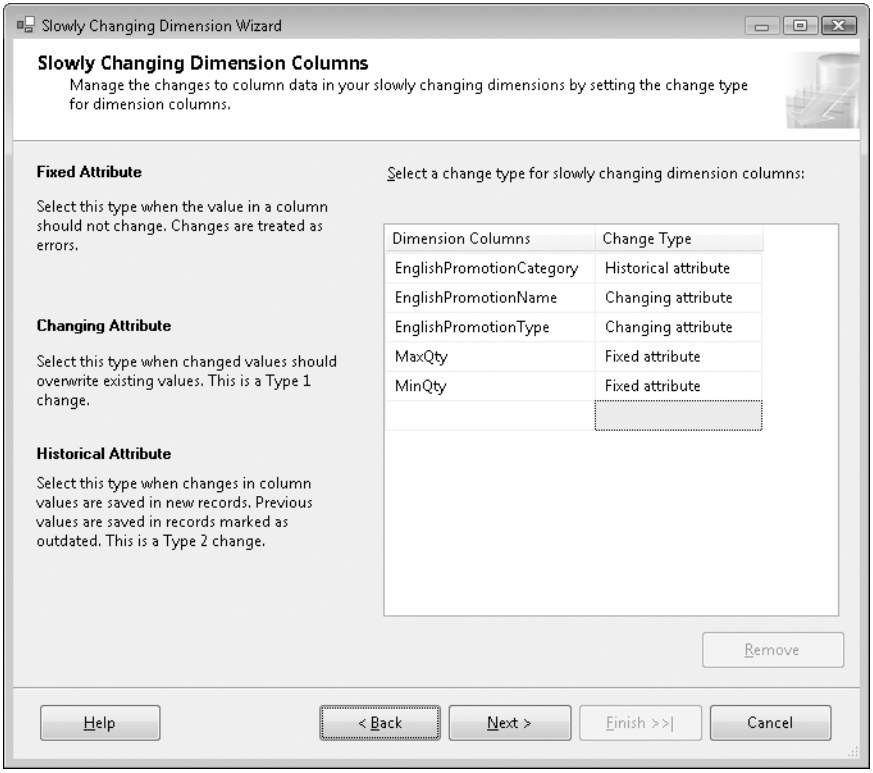


FIGURE 1-28 The Slowly Changing Dimension Wizard lets you define what kind of output should be created depending on the kind of change that occurs.

PRACTICE Creating Simple and Complex Data Flows

These exercises walk you through creating data flows that include sources, destinations, and one or more transformations. You will begin with a rather simple data flow but will then build a more complex data flow.

EXERCISE 1 Create a Simple Data Flow

In this exercise, you will develop a simple data flow that contains a source adapter, an Aggregate Transformation, and a destination adapter.

1. If necessary, start SQL Server Business Intelligence Development Studio (BIDS), open the project *TK 70-448 SSIS Project* you created in Lesson 2, "Creating and Editing Control Flow Objects," or open the completed exercise file from the companion CD, and then open the *MyPackage.dtsx* package for editing.
2. On the Control Flow tab of the SSIS Designer, drag Data Flow Task from the toolbox into the Sequence Container object. The Sequence Container object should now include an Execute SQL Task named Update ProductLine and a Data Flow Task object.

3. Drag the output arrow from the Update ProductLine Task onto the Data Flow Task object. The output arrow is green, which means it represents a precedence constraint; see Chapter 3 for more information about precedence constraints.
4. Click the Data Flow tab at the top of the SSIS Designer.
5. In the toolbox, drag OLE DB Source, located under the Data Flow Sources group, onto the data flow workspace. Right-click the OLE DB Source item and then click Edit to open the OLE DB Source Editor dialog box.
6. Select AdventureWorks2008 in the OLE DB Connection Manager list and then click OK.
7. From the Data Access Mode drop-down list, select SQL Command.
8. In the SQL Command text box, type the following query (available in the SQLCommandQuery.sql file in the ..\Source\Ch 01\ folder for the practice exercises):

```
SELECT SH.OrderDate, SD.LineTotal, P.ProductLine
FROM Sales.SalesOrderHeader SH
INNER JOIN Sales.SalesOrderDetail SD
ON SH.SalesOrderID = SD.SalesOrderID
INNER JOIN Production.Product P
ON SD.ProductID = P.ProductID
```

9. Click the Columns tab on the left, and then verify that the OrderDate, LineTotal, and ProductLine columns are shown as available columns in the source adapter.
10. Click OK in the OLE DB Source Editor dialog box.
11. From the Data Flow toolbox, drag an Aggregate Transformation onto the Data Flow design surface, just below the OLE DB Source adapter.
12. Link the OLE DB Source output to the Aggregate Transformation by dragging the green output arrow onto the Aggregate Transformation.
13. Edit the Aggregate Transformation by double-clicking it or by right-clicking it and then clicking Edit.
    a. In the Aggregate Transformation Editor dialog box, select OrderDate from the Input Column drop-down list, and then verify that the default operation Group By is selected for the new row.
    b. Add a second Input Column row by selecting the LineTotal column from the drop-down list. For the Operation column of the newly added LineTotal, select Sum from the list. And last, type **SubTotal** in the Output Alias column for the LineTotal row.
    c. Add a third Input Column row by selecting the ProductLine column from the list.
    d. Verify that the default operation Group By is selected for the new row.
    e. Click OK in the Aggregate Transformation Editor dialog box.
14. In the Data Flow toolbox, navigate to the Data Flow Destinations grouping of objects, and then drag the OLE DB Destination object onto the Data Flow design surface.

15. Connect the output of the Aggregate Transformation to the new OLE DB Destination object by dragging the output arrow from the Aggregate Transformation onto the OLE DB Destination adapter.
16. Right-click the OLE DB Destination adapter, and then click Edit to display the OLE DB Destination Adapter Editor dialog box.
    a. In the OLE DB Destination Adapter Editor dialog box, verify that the OLE DB Connection Manager drop-down list is set to AdventureWorks2008.
    b. Click the New button next to the Name Of The Table Or The View drop-down list.
    c. In the Create Table dialog box, change the name of the new table to **Sales_Summary**. The CREATE TABLE code listed in the window should look like the following:

```
CREATE TABLE [Sales_Summary] (
    [OrderDate] DATETIME,
    [SubTotal] NUMERIC (38,6),
    [ProductLine] NVARCHAR(2)
) ON [PRIMARY]
```

    d. Click OK in the Create Table dialog box.
    e. On the Mappings tab of the OLE Destination Editor dialog box, ensure that the columns are all mapped from source to destination.
    f. Click OK to save your settings.

    Figure 1-29 shows the completed data flow, with the source, aggregate, and destination components.

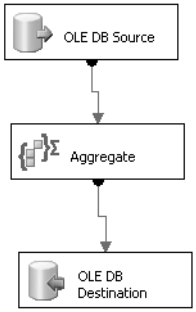


FIGURE 1-29 The data flow for this exercise contains an OLE DB Source adapter, an Aggregate Transformation, and an OLE DB Destination adapter.

17. Right-click the Data Flow design surface, and then click Execute Task. Observe the execution of the data flow to confirm successful completion of this exercise.
18. Click the Stop button on the toolbar to stop the debugger (or choose Debug, Stop Debugging from the Debug menu).
19. Click the Save All button on the BIDS toolbar.

EXERCISE 2 Create a Data Flow Destination

In this exercise, you will create a data flow that loads new records into the DimCustomer table of the AdventureWorksDW2008 database and that performs updates for existing records.

1. If necessary, start SQL Server Business Intelligence Development Studio (BIDS), open the project TK 70-448 SSIS Project, and then open the DimCustomer.dtsx package for editing. Your DimCustomer.dtsx package should contain an Execute SQL Task named Truncate Update Table, a Data Flow Task object, and a second Execute SQL Task named Batch Updates that was created in Lesson 2.
2. Click the Data Flow tab at the top of the SSIS Designer to navigate to the Data Flow design surface.
3. Drag an OLE DB Source adapter from the toolbox onto the design surface. Rename the OLE DB Source adapter **Customer Source**. Edit the source adapter and set the following properties as shown:

OLE DB Connection Manager	AdventureWorks2008
Data Access Mode	SQL Command

SQL Command Text
 	
(Code available in the CustomerSourceQuery.sql practice exercise file in the ..\Source\Ch 01\ folder.)
	`select convert(nvarchar(15),SC.`
	`AccountNumber) as CustomerAlternateKey,`
	`C.Title, C.FirstName, C.MiddleName,`
	`C.LastName, C.Suffix, C.EmailAddress,`
	`C.AddressLine1, C.AddressLine2,`
	`D.BirthDate, D.MaritalStatus,`
	`D.YearlyIncome, D.DateFirstPurchase,`
	`D.Gender, D.TotalChildren,`
	`D.NumberChildrenAtHome, D.Education,`
	`D.Occupation, D.HomeOwnerFlag,`
	`D.NumberCarsOwned`
	`from Sales.vIndividualCustomer C`
	`inner join Sales.Customer SC`
	`on C.BusinessEntityID = SC.PersonID`
	`inner join Sales.vPersonDemographics D`
	`on C.BusinessEntityID =`
	`D.BusinessEntityID`

4. Drag a second OLE DB Source adapter from the toolbox onto the Data Flow design surface, rename it to **Customer Dim**, and then edit it. Edit the source adapter and set the following properties as shown:

OLE DB Connection Manager	AdventureWorksDW2008
Data Access Mode	Table or view
Name Of The Table Or View	[dbo].[DimCustomer]

5. For this next set, you will be sorting the data from the sources on the business key. First, drag two Sort Transformations from the Data Flow toolbox onto the Data Flow design surface, and then connect the output arrow for the Customer Source adapter to the first Sort Transformation and the Customer Dim to the second Sort Transformation, as Figure 1-30 shows.

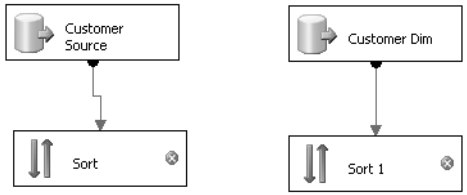


FIGURE 1-30 The initial data flow for this exercise contains two OLE DB Source adapters and two Sort Transformations.

6. Edit the first Sort Transformation and select the check box on the left side of the CustomerAlternateKey column in the Available Input Columns. Click OK to save the transformation.
7. Edit the second Sort Transformation and select the check box on the left side of the CustomerAlternateKey column in the Available Input Columns. Click OK to save the transformation.
8. From the Data Flow toolbox, drag a Merge Join Transformation to the design surface, and then connect the output arrow from the first Sort Transformation (originating from Customer Source) to the Merge Join Transformation. When prompted with the Input Output Selection dialog box, choose Merge Join Left Input from the Input drop-down list, and then click OK.

9. Also connect the output arrow of the second Sort Transformation (originating from Customer Dim) to the Merge Join Transformation.
10. Edit the Merge Join Transformation to display the Merge Join Transformation Editor dialog box.
    a. Change the Join Type drop-down list setting to Left Outer Join, which will retrieve all the rows from the originating Customer Source query (the left source of the Merge Join Transformation) and any matching rows from the right side (which is from the *dbo.DimCustomer* source).
    b. To return all the columns from the Customer Source query, select the check box immediately to the left of the Name column header in the left Sort list. Doing this will select all the check boxes for every column that is the desired result.
    c. In the right list of columns from Customer Dim, select only the check box next to the CustomerAlternateKey column.
    d. Scroll down the Output Columns list at the bottom of the Merge Join Transformation Editor dialog box to the very bottom, and for the CustomerAlternateKey column, change the Output Alias value to **Dim_CustomerAlternateKey**.
    e. Click OK to save the changes to the Merge Join Transformation.
11. From the Data Flow toolbox, drag a Conditional Split Transformation onto the Data Flow design surface, and then connect the output arrow from the Merge Join Transformation to the Conditional Split Transformation.
12. Edit the Conditional Split Transformation to display the Conditional Split Transformation Editor dialog box.
    a. Create a new output by typing **New Records** in the Output Name box for the first row of the output list.
    b. In the same row of the output list, type the following code in the Condition field:

```
ISNULL([Dim_CustomerAlternateKey]) == TRUE
```

    c. In the Default Output Name box, change the value from Conditional Split Default Output to **Updated Records**.
    d. Click OK to save your changes in the Conditional Split Transformation Editor dialog box.

13. From the Data Flow toolbox, drag an OLE DB Destination adapter to the Data Flow design surface (be sure not to drag the similar source adapter but rather the destination adapter), and then change its name to **DimCustomer Table**.
14. Drag the output arrow of the Conditional Split Transformation onto this new OLE DB Destination adapter. When prompted in the Input Output Selection dialog box, select New Records from the Output drop-down list, and then click OK.
15. Right-click the DimCustomer Table Destination adapter that you just created and click Edit to display the OLE DB Destination Editor dialog box. Set the following properties in the OLE DB Destination Editor dialog box:

OLE DB Connection Manager	AdventureWorksDW2008
Data Access Mode	Table Or View—Fast Load
Name Of The Table Or View	[dbo].[DimCustomer]

    a. While you are still in the OLE DB Destination Editor dialog box, click the Mappings tab in the left area of the dialog box. This automatically maps the columns from the data flow to the DimCustomer table based on column name and data type.
    b. Not all columns will be mapped. From the Available Input Columns list, locate the Education column and drag it on top of the EnglishEducation column of the Available Destination Columns list. Do the same for Occupation to EnglishOccupation and HomeOwnerFlag to HouseOwnerFlag.
    c. Click OK to save your changes in the OLE DB Destination Editor dialog box.
16. Add a second OLE DB Destination adapter to the Data Flow design surface, and then connect another output arrow from the Conditional Split Transformation to the new OLE DB Destination adapter. Rename the destination adapter **DimCustomer Update Table**.
17. Edit the DimCustomer Update Table destination adapter you just created to display the OLE DB Destination Editor dialog box. Set the following properties in the OLE DB Destination Editor dialog box:

OLE DB Connection Manager	AdventureWorksDW2008
Data Access Mode	Table Or View—Fast Load
Name Of The Table Or View	[dbo].[stgDimCustomerUpdates] (This table was created in Lesson 2.)

a. While you are still in the OLE DB Destination Editor dialog box, click the Mappings tab. This automatically maps the columns from the data flow to the DimCustomer table based on column name and data type.

b. Not all columns will be mapped. From the Available Input Columns list, locate the Education column and drag it on top of the EnglishEducation column of the Available Destination Columns list. Do the same for Occupation to EnglishOccupation and HomeOwnerFlag to HouseOwnerFlag.

c. Click OK to save your changes in the OLE DB Destination Editor dialog box.

Your data flow should now resemble the one shown in Figure 1-31. You can find the completed exercises in the ..\Source\Ch 01\ folder of the Training Kit materials.

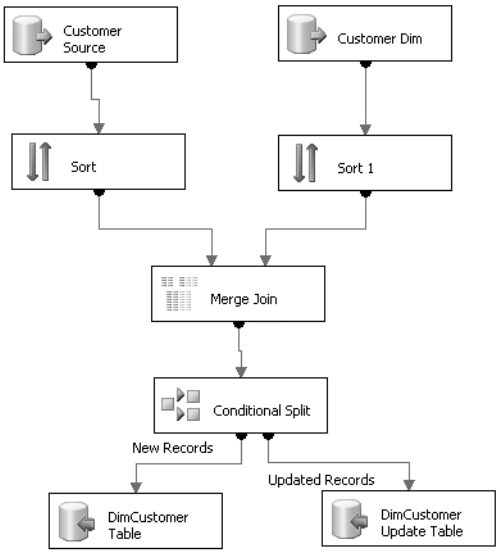


FIGURE 1-31 The final data flow for this exercise contains several sources and destinations, with transformation logic to handle inserts and to stage updates for the DimCustomer table.

18. Confirm the correct development of your package by executing the package in BIDS.

19. Choose Debug, Stop Debugging from the Debug menu to stop the debugger, and then click the Save All button on the BIDS toolbar.

Quick Check

1. How would you use SSIS to import a file from an FTP server to a SQL Server table?
2. You need to migrate a user-created Access database to SQL Server, but the Data Flow toolbox does not contain an Access source adapter. How do you import this data into SQL Server?
3. The Multicast Transformation and the Conditional Split Transformation both can have multiple outputs. Which transformation would you use if you needed to send rows matching certain conditions out one output and rows matching different conditions out another?
4. Describe the transformations you could use to combine data from two different database tables that exist on two different servers.

Quick Check Answers

1. First, you would use an FTP Task to copy the file to the machine on which SSIS is installed. You can then import the file into a SQL Server table by using a Data Flow Task configured with a Flat File Source adapter and either a SQL Server Destination adapter or an OLE DB Destination adapter.
2. Although not listed in the toolbox, Access is one of the many database sources and destinations that SSIS works with. To extract data from Access, you first need to make a package connection to the Microsoft Jet OLE DB Provider. You can then use the OLE DB Source adapter to select the table or perform a custom query.
3. The Conditional Split Transformation lets you define expressions against which the rows from the source are evaluated. For every row, the expressions are evaluated in order, and a row is sent out the first output when the matching expression evaluates to True. Therefore, any single row can go out only one output. With a Multicast Transformation, on the other hand, all rows go out every output.
4. To combine data from two different database tables that exist on two different servers, you could use the Merge Join Transformation, which combines datasets by joining the rows across a set of common keys. This transformation allows an inner join, a left outer join, or a full outer join. You could also use a Lookup Transformation to associate data from two sources. The Lookup can cache a table in memory and, through matching columns, can return new columns to the data flow.

Case Scenario: Creating an ETL Solution

The business development department of Adventure Works has requested that you implement a data mart that it can use to analyze reseller sales against salesperson sales targets. Your first task is to create a series of SSIS packages that move data from the source Enterprise Resource Planning (ERP) system to a data mart database that contains fact tables and dimension tables.

1. How would you work within BIDS to create SSIS project structures, packages, project data sources, and package connections?
2. What transformations would you use, and how would you implement the data flow that loads dimension tables?
3. What transformations would you use, and how would you implement the data flow that loads fact tables?

Chapter Summary

- Creating SSIS packages involves working with BIDS and creating a new SSIS project.
- The main object in an SSIS project is a package, which contains the business logic to manage workflows and process data.
- Within a package, the control flow lets you create tasks and containers, which provide the ability to run process-oriented operations.
- The Data Flow Task is the second core object (behind the control flow) in an SSIS package, enabling data-processing operations.
- The data flow uses source adapters, destination adapters, and transformations.

CHAPTER 6

Extending SSAS Cubes

You can extend SQL Server 2008 Analysis Services (SSAS) cubes in versatile ways to meet advanced business requirements. You can define attribute relationships and hierarchies to optimize the cube design and facilitate data analytics. In addition, you can further enrich the reporting experience by building an end-user layer consisting of key performance indicators (KPIs), actions, translations, and perspectives. You can also use Multidimensional Expressions (MDX) language expressions to define important business metrics not available in the fact tables or that require custom expressions.

This chapter builds on the Adventure Works cube implemented in Chapter 5, "Developing SSAS Cubes." The source code for the Chapter 6 practices can be installed from the companion CD. This chapter starts by discussing attribute and dimension relationships. You will learn how to extend the Unified Dimensional Model (UDM) with KPIs, actions, translations, and perspectives. And after describing the MDX query fundamentals, the chapter will explore MDX calculated members and named sets.

Exam objectives in this chapter:

- Implement dimensions in a cube.
- Configure dimension usage in a cube.
- Implement custom logic in a cube by using MDX.

Before You Begin

To complete this chapter, you must have:

- Knowledge of dimensional modeling and SSAS cubes.
- An understanding of SSAS dimensions and measures.
- Practical experience with Analysis Services projects in Business Intelligence Development Studio (BIDS).

Lesson 1: Defining User Hierarchies and Dimension Relationships

Estimated lesson time: 60 minutes

From an analytical standpoint, a *dimension* gives users the ability to analyze data by subject areas. Dimensions let users isolate, drill down, roll up, categorize, filter, summarize, and perform other actions on data. In the UDM, a dimension is a logical container of attributes. Understanding how dimension attributes relate to each other will help you optimize the UDM performance.

Defining Attribute Relationships

An SSAS dimension gets its data from one or more dimension tables that reside in the data source. If the dimension is based on a star schema, its source is a single dimension table. Dimensions based on a snowflake database schema typically span multiple dimension tables. In the Adventure Works cube, for example, the Product dimension is built on a snowflake schema that includes the DimProduct, DimProductSubcategory, and DimCategory tables. The rest of the dimensions are of a star type and use single dimension tables.

Unlike a relational database, which stores data in two-dimensional structures of rows and columns, the UDM is a multidimensional system that supports data hierarchies—such as the hierarchy formed by Year, Quarter, and Month levels—and the relationships that exist among the attributes forming these hierarchies. Table 6-1 summarizes such relationships for a subset of the attributes in the Product dimension you designed in Chapter 5.

TABLE 6-1 Understanding Attribute Relations

TABLE COLUMN	ATTRIBUTE TYPE	RELATIONSHIP TO PRIMARY KEY	OTHER RELATIONSHIPS
ProductKey	Primary key	Dimension key	
Description	Attribute	1:1	
Color	Attribute	Many:1	
Subcategory	Attribute	Many:1	Many:1 with Category
Category	Attribute	Many:1	

All dimension attributes are directly or indirectly related to the dimension key attribute with one-to-one (1:1) or many-to-one (Many:1) logical relationships. For example, the Description attribute has a 1:1 relationship with the dimension key attribute (ProductKey), assuming that each product has a unique description. In contrast, the Color attribute has a Many:1 relationship to the dimension key because multiple products can have the same color. Similarly, the Subcategory attribute has a Many:1 relationship with the dimension key because one subcategory can include multiple products. The Subcategory attribute has a logical Many:1 attribute with the Category attribute because one product category can include multiple subcategories.

BEST PRACTICES **DEFINE ATTRIBUTE RELATIONSHIPS**

By default, the UDM is not aware of the logical relationships among attributes. It defines Many:1 relationships between all attributes and the dimension key, except with snowflake dimensions, where it automatically creates separate Many:1 relationships among attributes from different tables. As a modeler, you must spend additional time to understand and define appropriate attribute relationships.

Setting up relevant attribute relationships provides the following benefits:

- SSAS can effectively store data. The server can optimize the storage of the attribute hierarchies.
- SSAS can effectively retrieve and aggregate data. In the absence of attribute relationships, the server must query the fact data and then group data by the attribute. For example, if there are no attribute relationships between the Subcategory and Category attributes, a query that groups data by product category will scan the fact data on the fly to summarize the individual products. However, if the attributes are related, the server could optimize the query and use the subcategory subtotals.

To learn more about the importance of defining attribute relationships, read the white paper "Microsoft SQL Server 2008 Analysis Services Performance Guide" (see References).

Compared with SSAS 2005, SSAS 2008 has simplified the process of analyzing and defining attribute relationships by introducing a new Attribute Relationships tab in the Dimension Designer, as Figure 6-1 shows.

This tab has a Diagram pane, an Attributes pane, and an Attribute Relationships pane. If the Attributes and Attribute Relationships panes are not visible, click the Show List Views button on the Dimension Designer toolbar on the Attribute Relationships tab.

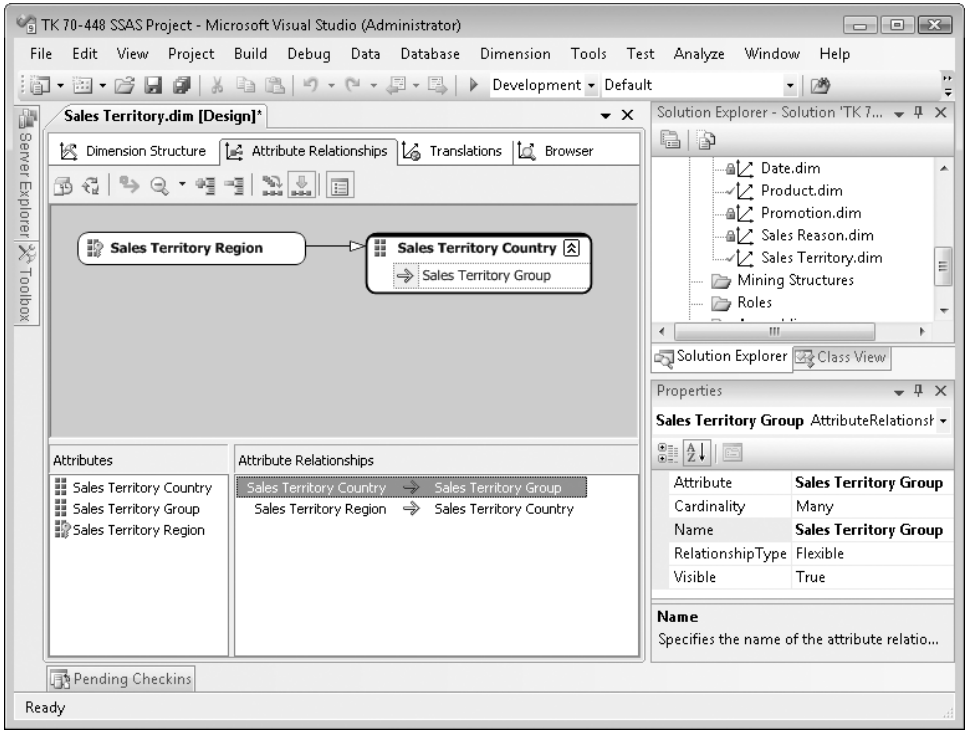


FIGURE 6-1 Use the Attribute Relationships tab in the Dimension Designer to view and change attribute relationships.

The Diagram pane shows an attribute relationship graph so that you can quickly visualize existing relationships. The Attributes pane shows the dimension attributes. And the Attribute Relationships pane lists existing attribute relationships. To select an attribute relationship, click the relationship in the Attribute Relationships pane or select the arrow that connects the attributes in the Diagram pane, and then use the Properties pane to modify the relationship properties. Table 6-2 describes the relationship properties.

TABLE 6-2 Attribute Relationship Properties

PROPERTY	PURPOSE
Attribute	Specifies the name of the related attribute.
Cardinality	Specifies the cardinality of the relationship between the two attributes (could be One or Many).
Name	Specifies the relationship name. This translates to the name of the member property that the end user will see when browsing the attribute in a browser that supports member properties, such as Microsoft Office Excel 2007.

PROPERTY	PURPOSE
RelationshipType	Specifies whether the relationship can change over time. Set this property to Flexible when you expect changes, such as for a product that can change a subcategory. Set it to Rigid if the relationship does not change, such as for a relationship between the Quarter and Year attribute in a Time dimension.
Visible	Defines the visibility of the member property. When set to False, the corresponding member property will not be shown to the end user.

To create a new attribute relationship, follow these steps:

1. In the Attributes pane, right-click the source attribute that is on the "one" side of the relationship, and then select New Attribute Relationship.
2. Configure the relationship by using the Create Attribute Relationship dialog box, shown in Figure 6-2.

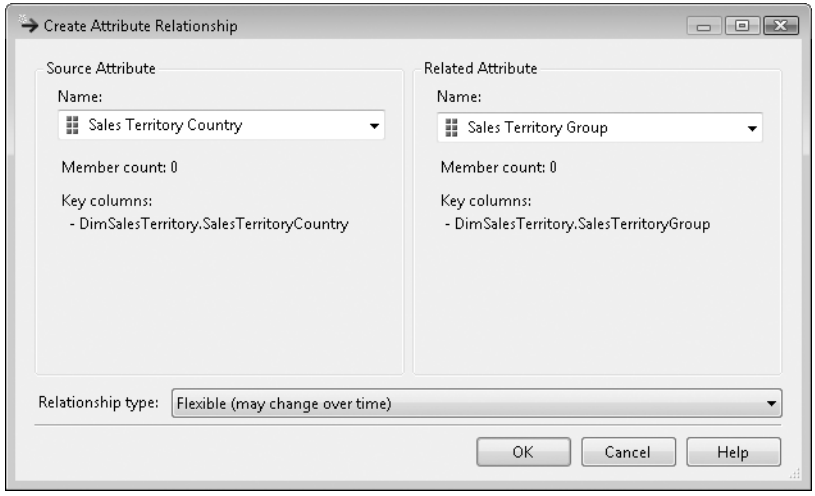


FIGURE 6-2 Use the Create Attribute Relationship dialog box to set up a new attribute relationship.

Make sure that the Name drop-down list below Source Attribute shows the attribute that is on the "many" side of the relationship and that the Name drop-down list below Related Attribute shows the attribute on the "one" side of the relationship. Use the Relationship Type drop-down list to specify a Flexible or Rigid relationship type. You can also create a new relationship in the Dimension Designer by dragging the source attribute onto the related attribute in the Diagram pane. To delete an attribute relationship, select the relationship in the Attribute Relationships pane (or click the arrow connector in the Diagram pane) and then press Delete.

Creating and Modifying User Dimension Hierarchies

The UDM is an attribute-based model in which the cube space is defined by attribute hierarchies. In Chapter 5, you saw that each dimension attribute forms an attribute hierarchy whose members are formed by the distinct values of the attribute source column. For example, the members of the Sales Territory Country attribute hierarchy contain the distinct countries stored in the SalesTerritoryCountry column. Attribute hierarchies are incredibly useful because they let end users analyze data aggregated by the members of the hierarchy. As a modeler, you should consider setting up additional user hierarchies that combine attributes and define useful navigational paths.

For example, many users who interact with cubes through Excel or another user application need to be able to drill down into a hierarchy that provides expanding levels of detail. Such hierarchies might include the year, quarter, month, and day within a Date dimension; the category, subcategory, and product name within a Product dimension; or the country, state, and city within a Geography or Customer dimension. SSAS supports dimensions that have multiple user hierarchies, giving end users multiple options for exploring cube data.

You define hierarchies within the context of a given dimension by using the Dimension Structure tab of the Dimension Designer. The Hierarchies pane in Figure 6-3 shows a Sales Territory user hierarchy that has Group, Country, and Region levels.

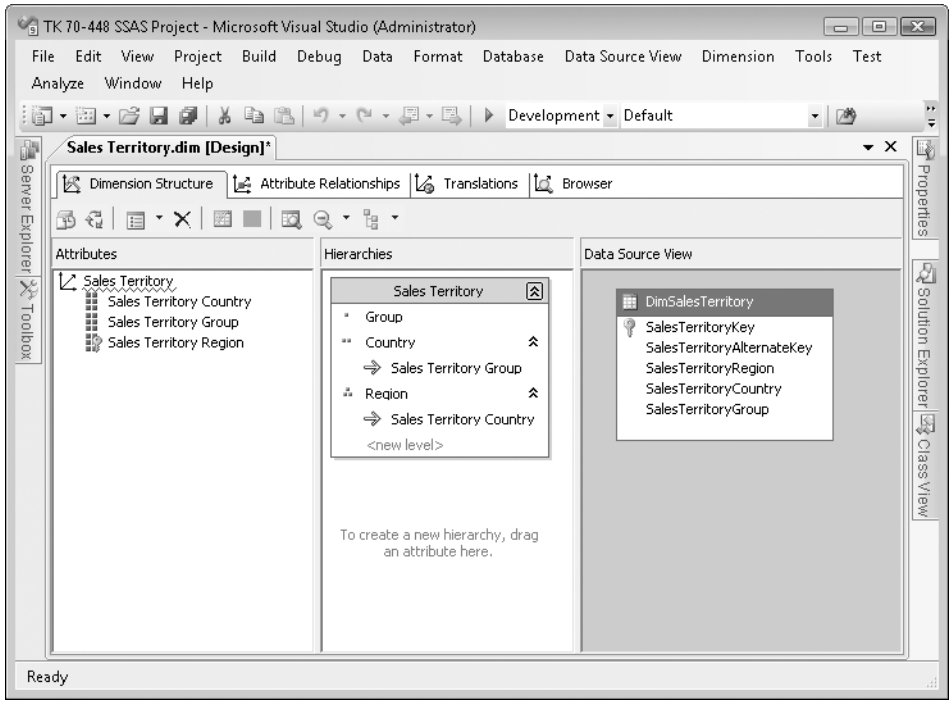


FIGURE 6-3 The Sales Territory user hierarchy lets end users drill down into the data by Group, Country, and Region.

You define hierarchies by following these steps:

1. To add a new hierarchy within a dimension, right-click the attribute that will be the top level, and then select Start New Hierarchy. For example, in the Sales Territory dimension, which you implemented in Chapter 5, you can create a new hierarchy by selecting the Sales Territory Group attribute as the top level. Alternatively, you can drag an attribute to the Hierarchies pane to serve as the top level of a new hierarchy.
2. To add additional levels, right-click each attribute that will serve as a level in the hierarchy, and then select Create Level. Alternatively, to create a new level, you can drag the attribute to the <new level> placeholder within the hierarchy's structure in the Hierarchies pane. You can add more hierarchies to the dimension's design by repeating this procedure.

After you have created a new hierarchy, you can use the Properties window to modify the properties for the hierarchy and each of its levels. For example, you can use the DisplayFolder property of the hierarchy to place it within the display folder used for attribute hierarchies or a different folder. Or you can use the Name property to change the name of a level from the name of the attribute used as the source for the level to a name that makes more sense from an end user's perspective. Alternatively, you can rename a level in place in the Hierarchies pane.

EXAM TIP

A user hierarchy typically assumes Many:1 relationships among its levels. For example, a territory group can have many countries, and a country can have many regions. Although you can set up user hierarchies without defining attribute relationships, you should avoid this structure because it results in a poor dimension design. When the Dimension Designer detects missing attribute relationships, it displays a warning icon next to the name of the hierarchy in the Hierarchies pane. You can expand the hierarchy levels in the Hierarchies pane to see the existing attribute relationships. To optimize your dimension design, switch to the Attribute Relationships pane and set up Many:1 attribute relationships. Also consider hiding attributes that are used in user hierarchies by setting their AttributeHierarchyVisible property to False to prevent redundancy in the user experience.

Associating Dimensions to Measure Groups

When you add a dimension to a cube, the Cube Designer determines which measure groups the dimension is related to by using the table relationships metadata within the data source view (DSV) on which the cube is based. In most cases, the associations the Cube Designer creates are correct. However, you can use the Dimension Usage tab to review the relationships between dimensions and measure groups and to correct any problems.

BEST PRACTICES **DEFINE DIMENSION USAGE WITHIN A CUBE**

Within SSAS, a cube can contain multiple measure groups and multiple dimensions. Although many of the measure groups are likely related to the same dimensions, not every dimension will be related to every measure group. In addition, SSAS supports different types of measure group-to-dimension relationships to support different database modeling requirements. You can use the Dimension Usage tab within the Cube Designer to define the various relationships that should exist between each measure group and dimension within the cube.

Figure 6-4 shows the Dimension Usage tab within the Cube Designer, which displays along the top of the table the list of measure groups in the cube and along the left side of the table the list of dimensions in the cube.

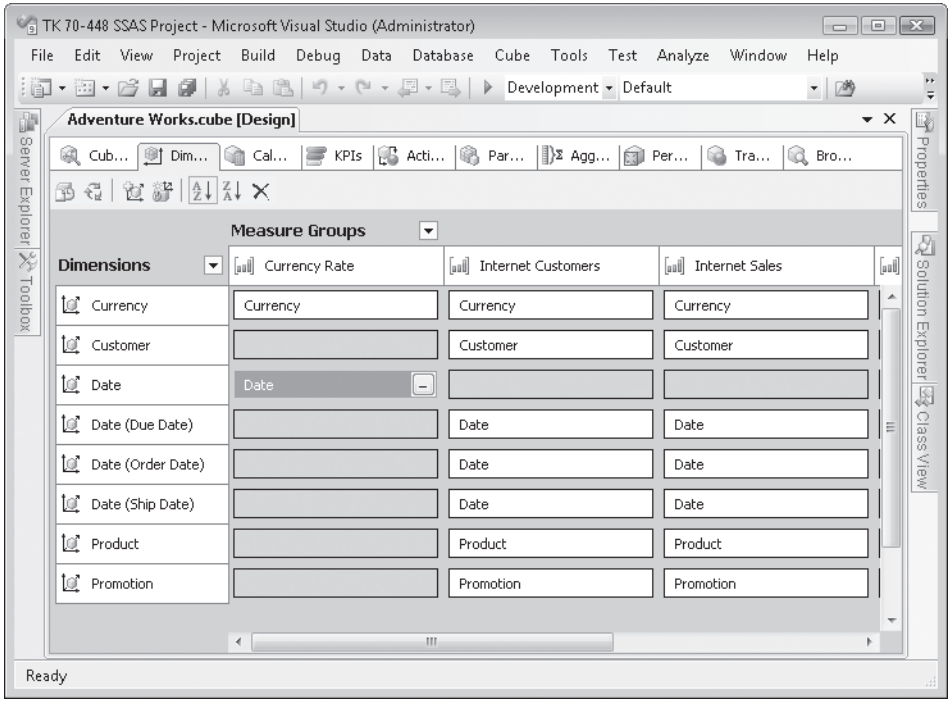


FIGURE 6-4 The Dimension Usage tab shows the relationships between a cube's measure groups and its dimensions.

You can sort the dimensions and measure groups alphabetically in ascending or descending order by clicking the appropriate toolbar button. The intersection of a given measure group and dimension defines the relationship between the two. If a dimension is not related to a measure group, the intersection will show an empty gray cell.

In Figure 6-4, the Date dimension joins only the Currency Rate measure group. This is a redundant Date dimension relationship that the Cube Designer defined when you added the Currency Rate measure group. To optimize the cube design and reduce the number of dimensions, consider deleting the Date dimension and reusing an existing role-playing Date dimension, such as Date (Order Date), to browse the Currency Rate data.

Selecting Relationship Types

If you select a cell within the table on the Dimension Usage tab, an ellipsis (...) button provides access to the Define Relationship dialog box, shown in Figure 6-5. You can use this dialog box to create or modify a relationship between a dimension and a measure group. The dialog box lets you select the relationship type and set various properties related to the relationship.

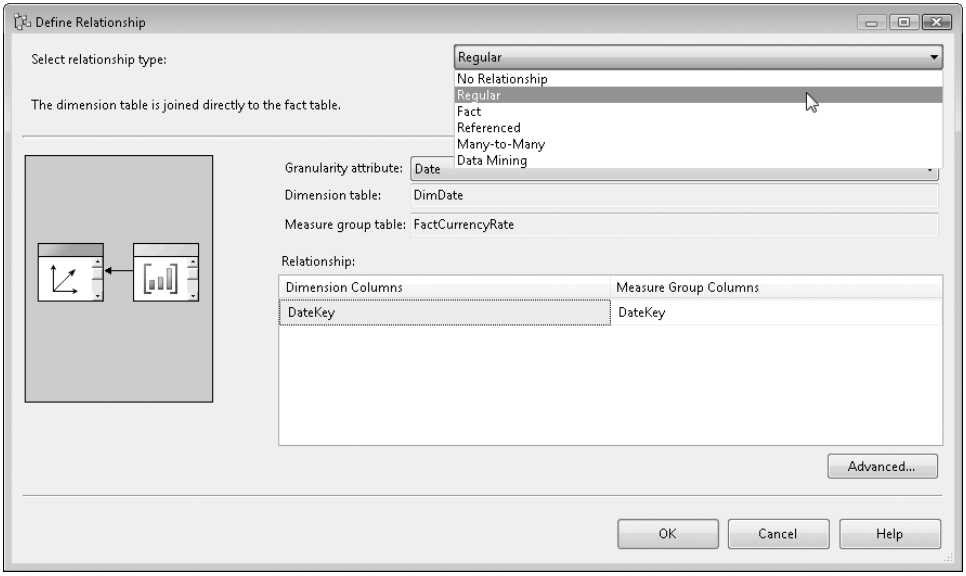


FIGURE 6-5 In the Define Relationship dialog box, you can add new relationships and modify existing ones.

When relating a given dimension to a measure group, you can select one of several types of relationships, depending on your specific need. Your choice will be driven primarily by the database design and model you are using to define the cube. Table 6-3 describes all the relationship types you can define in a cube.

TABLE 6-3 Dimension Usage Relationship Types

RELATIONSHIP TYPE	PURPOSE
Regular	Defines the relationship when a dimension is joined directly to a measure group through a specific attribute called the "granularity" attribute.
Fact	Used when the dimension is based on the fact table used to define the measure group.
Referenced	Used when a given dimension is related to a measure group through an intermediate dimension.
Many-To-Many	Specifies that a dimension is related to a given measure group through an intermediate measure group.
Data Mining	Defines the relationship between a dimension based on a data mining model and a given measure group.

EXAM TIP

Sometimes after analyzing the database schema and designing fact and dimension tables, you will be left with columns in the fact table that do not justify moving them to designated dimension tables, such as Sales Order Number. Yet reporting requirements might require you to let end users browse data by these columns—for example, to see all sales order line items for a given sales order. You can meet such requirements by building dimensions directly from the fact table and then joining them to the related measure groups through a fact relationship.

To select a given relationship type, open the Define Relationship dialog box, and then select the relationship type from the Select Relationship Type drop-down list at the top of the dialog box. Based on the relationship type you select, the other options available within the dialog box will change to reflect the required settings for that relationship type. Ensure that the appropriate values are entered for the options shown, and then click OK to close the dialog box.

A dimension does not need to join a measure group at the dimension key level. For example, the AdventureWorksDW2008 database has a FactSalesQuota fact table that stores the quarterly sales quota of salespersons. At the same time, the lowest level of the DimDate dimension table is days. However, SSAS supports joining DimDate to FactSalesQuota at the quarter level. When defining a relationship at a higher level, it is extremely important to define appropriate Many:1 relationships. Failure to do so might yield incorrect subtotals.

PRACTICE Creating and Modifying Dimension Hierarchies

The following exercises will familiarize you with the tasks related to defining attribute relationships, working with dimension hierarchies, and creating relationships between dimensions and measure groups. When beginning the exercises, use the Start Here project from the Documents\Microsoft Press\MCTS Training Kit 70-448\Source\Ch 06\TK448-ch06 Start Here\ folder in the installed practice files.

EXERCISE 1 Define Attribute Relationships

In this exercise, you will define attribute relationships in the Date, Product, and Sales Territory dimensions.

1. In Solution Explorer, double-click the Date dimension to open it in the Dimension Designer.
2. Click the Attribute Relationships tab. Notice that the Diagram pane shows the Date dimension key attribute. All other attributes are related to the dimension key through Many:1 relationships; there are no relationships among the nonkey attributes.

   However, some attributes have logical Many:1 relationships with other attributes. For example, the Month Name attribute has a Many:1 relationship with the Calendar Quarter attribute, and the Calendar Quarter attribute has a Many:1 relationship with Calendar Year. Next, you will change the dimension design to reflect these relationships.
3. Right-click the Month Name attribute, and then select New Attribute Relationship. In the Create Attribute Relationship dialog box that appears, make sure that the related attribute is Calendar Quarter. Change the Relationship Type to Rigid (Will Not Change Over Time) because months cannot change quarters. Click OK.
4. The Month Name attribute also has a Many:1 relationship with fiscal quarters. Create a second Rigid relationship between the Month Name attribute and the Fiscal Quarter attribute.
5. Create a Rigid relationship between the Calendar Quarter and Calendar Year attributes and another Rigid relationship between the Fiscal Quarter and Fiscal Year attributes. Save your changes, and then close the Dimension Designer.
6. In Solution Explorer, double-click the Product dimension to open it in the Dimension Designer.
7. Click the Attribute Relationships tab, and notice that there are already attribute relationships between the Product and Subcategory attributes and the Subcategory and Category attributes. The Dimension Designer discovered that the Product dimension has a snowflake schema and automatically generated these attribute relationships.

Notice in the Attributes pane of the Attribute Relationships tab that warning icons appear in some attribute relationships to warn you that the relationship and related attribute names differ. Consequently, when end users browse the cube, they will see a member property that has the relationship name instead of the name of the related attribute.

8. In the Attribute Relationships pane, select the Product–Description relationship, and, in the Properties window, clear its Name property. If you do not specify the Name property, it defaults to the Attribute property.
9. Repeat step 8 for the Product–Subcategory and Subcategory–Category relationships. Save your changes, and then close the Dimension Designer.
10. In Solution Explorer, double-click the Sales Territory dimension to open it in the Dimension Designer.
11. Click the Attribute Relationships tab, and set up a Many:1 Flexible relationship between the Sales Territory Country and Sales Territory Group attributes. Save your changes, and then close the Dimension Designer.

EXERCISE 2 Define User Hierarchies

In this exercise, you will define user hierarchies in the Date, Product, and Sales Territory dimensions. These user hierarchies will provide logical navigational paths to explore the cube data.

1. In Solution Explorer, double-click the Date dimension to open it in the Dimension Designer. Currently, the Date dimension does not have any user hierarchies defined, but end users often browse data by year, quarter, month, and day. In this exercise, you will implement a Calendar user-defined hierarchy that includes these levels.
2. Click the Dimension Structure tab. Drag the Calendar Year attribute from the Attributes pane to the Hierarchies pane to start a new user hierarchy where the top level is the Calendar Year attribute.
3. Click the new hierarchy to select it. Rename the hierarchy by changing its Name property to **Calendar**.
4. Drag the Calendar Quarter attribute from the Attributes pane onto the <new level> placeholder in the Calendar hierarchy. This creates the second hierarchy level, which will let the user drill from years to quarters.
5. Repeat step 4 twice more to add the Month Name and Date levels to the Calendar hierarchy.

   Note that if you skipped Exercise 1, the Dimension Designer will show a warning indicator in the hierarchy caption. When you point to this indicator, the Dimension Designer will display the following tooltip:

   Attribute relationships do not exist between one or more levels of this hierarchy. This may result in decreased query performance.

   If this happens, follow the steps in Exercise 1 to set up attribute relationships.

6. Create a new Fiscal user hierarchy that contains the Fiscal Year, Fiscal Quarter, Month Name, and Date levels. Save your changes, and then close the Dimension Designer.
7. In Solution Explorer, double-click the Product dimension to open it in the Dimension Designer.
8. Define a Product Categories user hierarchy that contains Category, Subcategory, and Product levels. Save your changes, and then close the Dimension Designer.
9. In Solution Explorer, double-click the Sales Territory dimension to open it in the Dimension Designer.
10. Define a new user hierarchy that contains Sales Territory Group, Sales Territory Country, and Sales Territory Region levels.
11. Rename the hierarchy to **Sales Territories**. In addition, rename the Sales Territory Group level to **Group,** the Sales Territory Country level to **Country**, and the Sales Territory Region level to **Region**. Save your changes, and then close the Dimension Designer.

EXERCISE 3 Review and Modify Dimension Usage Relationship Types

In this exercise, you will create an Internet Sales Order Details fact dimension and set up a Fact relationship between this dimension and the Internet Sales measure group. You will also define a many:many dimension relationship. Last, you will optimize the Adventure Works cube by removing redundant dimension relationships.

Sometimes you might need to create a dimension directly from columns that exists in the fact table. For example, the FactInternetSales fact table includes SalesOrderNumber and SalesOrderLineNumber columns. Suppose that you need to support detail reports that will show sales orders and their associated sales items. Because there is not a designated Sales Order dimension table, you can create a Fact dimension (also called degenerate) directly from the fact table.

1. Open the Adventure Works DW2008 DSV, and then add a named calculation called LineItemDescription to the FactInternetSales table that uses the following expression:

```
CONVERT ( CHAR ( 10 ), SalesOrderNumber )  + 'Line '  +
CONVERT ( CHAR ( 4 ), SalesOrderLineNumber )
```

2. In Solution Explorer, right-click the Dimensions folder, and then select New Dimension. On the Select Creation Method page of the Dimension Wizard, accept the Use An Existing Table option.
3. On the Specify Source Information page, expand the Main Table drop-down list, and then select FactInternet Sales. Expand the Name Column drop-down list, and then select the LineItemDescription column.
4. On the Select Related Tables page, clear the check boxes for all suggested related tables.

5. On the Select Dimension Attributes page, clear the check boxes for all attributes except Sales Order Number.
6. On the Completing The Wizard page, name the new dimension **Internet Sales Order Details**, and then click Finish.
7. In Solution Explorer, double-click the Adventure Works cube to open it in the Cube Designer. On the Cube main menu, select Add Cube Dimension. In the Add Cube Dimension dialog box, select the Internet Sales Order Details dimension, and then click OK.
8. Click the Dimension Usage tab, and then review the existing dimension relationships. Notice that the Internet Sales Order Details dimension is related to the Internet Sales measure group through a Fact relationship. That is because the dimension is based on the same fact table that the measure group is based on.

   Notice too that the Sales Reason dimension does not join the Internet Sales measure group because the intersecting cell is empty. In the AdventureWorksDW2008 database, a sales order can be associated with one or more sales reasons, such as Promotion or Marketing, and a sales reason can be associated with one or more sales orders. The FactInternetSalesReason table represents the Many:Many relationship between sales reasons and orders. SSAS supports many-to-many dimension relationships.
9. Click the ellipsis (…) button in the intersecting cell between the Sales Reason dimension and the Internet Sales measure group.
10. In the Define Relationships dialog box that appears, expand the Select Relationship Type drop-down list, and then select Many-To-Many. Expand the Intermediate Measure Group drop-down list, and then select the Internet Sales Reason measure group whose source fact table is FactInternetSalesReason. Click OK to create the many-to-many dimension relationship.

    Notice that the Date dimension joins only the Currency Rate measure group. As it stands, this relationship is not useful because you cannot browse the data in the other measure groups by this dimension. You can optimize the cube design by deleting the Date dimension and reusing one of the existing Time role-playing dimensions, such as Date (Order Date).

11. Right-click the Date dimension in the Dimensions column, and then select Delete to remove the Date dimension. Click OK in the Delete Objects dialog box that appears, to confirm the deletion.
12. You will use the Date (Order Date) dimension to browse the data in the Currency Rate measure group. Click the ellipsis (…) button in the intersecting cell between the Date (Order Date) dimension and the Currency Rate measure group.
13. In the Define Relationship dialog box, expand the Select Relationship Type drop-down list, and then select Regular.
14. Expand the Granularity Attribute drop-down list, and then select the Date attribute, because the Date (Order Date) dimension will join the Currency Rate measure group at the date level.
15. Expand the drop-down list in the Measure Group Columns column in the Relationship grid, and then select the DateKey column. Click OK.
16. You can rename any cube dimension. Because the Date (Order Date) dimension now fulfills a more generic role, you will change its name to Date. Click the Date (Order Date) dimension in the Dimension column, and then rename it in place to **Date**. Save your changes.
17. In Solution Explorer, right-click the project node, and then select Deploy to deploy the changes and process the Adventure Works cube. If a message stating that the server content is out of date appears, click Yes to build and deploy the database.
18. Optionally, use the Cube Browser tab to test the changes. For example, explore the measures in the Internet Sales measure group by the Sales Reason dimension. Notice that the aggregate for the Other sales reason type correctly handles the fact that many sales reasons can be selected for a single Internet order (that is why you need to create a many-to-many relationship between the dimension and the measure group).

Quick Check

1. Why should you spend time defining appropriate attribute relationships?
2. When creating a dimension, can you create different hierarchies to represent every possible combination of attributes and to maximize the options available to end users for using the hierarchies to explore cube data?
3. Can you create hierarchies directly from columns within a dimension's table?
4. Can a dimension be related to a measure group if the underlying dimension table is not related to the appropriate fact table in a primary key–to–foreign key relationship?
5. Must every dimension you add to a cube be related to every measure group within that cube?

Quick Check Answers

1. Proper attribute relationships optimize storage and improve query performance because the server might be able to produce the totals from the related attribute totals instead of scanning the fact table.
2. Although technically you can create different hierarchies to represent every combination of attributes, a large number of hierarchies within a dimension design will likely offer too many options and confuse end users. Generally, users can create their own hierarchies by simply nesting (cross-joining) different attributes onto the rows or columns of a given query, although this capability is somewhat dependent on the applications they are using. So having attributes available only for the most commonly requested or needed hierarchies is probably your best design strategy.
3. No, you cannot create hierarchies directly from columns within a dimension's table. Hierarchies can be created only based on attributes that have been added to the dimension's design.
4. Although the dimension cannot be related to the measure group in a Regular relationship, you might be able to create a Referenced or Many-To-Many relationship if an intermediate dimension table or intermediate fact table related to the dimension and measure group in question is available. This capability within SSAS provides an elegant solution to various database modeling requirements.
5. No, you do not have to relate every dimension you add to a cube to every measure group in the cube. In fact, you can create a cube that includes multiple measure groups whose source fact tables are related to different sets of dimensions. This lets end users browse the data in a way that makes sense from a business perspective rather than forcing them to analyze data in a way that is constrained by the underlying database design.

Lesson 2: Creating KPIs, Actions, Translations, and Perspectives

Estimated lesson time: 75 minutes

Many organizations use key performance indicators (KPIs) to gauge their business performance. KPIs are quantifiable measures that represent critical success factors, and analysts use them to measure company performance, over time, against a predefined goal. For example, Sales Profit, Revenue Growth, and Growth In Customer Base are good KPI candidates. KPIs are typically used as part of a strategic performance measurement framework, commonly known as a *business scorecard*.

REAL WORLD

Teo Lachev

KPIs are typically shown on a *dashboard* page. A digital dashboard, also known as an enterprise dashboard or an executive dashboard, is a business intelligence (BI) tool that helps track the status (or health) of a company via KPIs.

One of my recent projects involved building a Microsoft Office SharePoint–based dashboard page for displaying various KPIs—such as Return Of Assets, Growth In Customer Base, and Percentage Of Customers Profitable—that were defined in an SSAS cube. What makes SSAS suitable for KPIs is that the server automatically calculates the KPI properties as the user slices the cube data. For example, the user could drill down into the Time dimension, and the cube would calculate the KPIs across the hierarchy at year, quarter, and month levels.

In addition, we benefited from the supreme performance that SSAS offers. Besides showing KPIs for the current and previous periods, our dashboard page included several charts and report views that showed historical trends. The page would submit many queries to the SSAS server, and some of the queries would request a large number of calculated measures across many time periods. But by carefully fine-tuning the cube performance, we were able to render the page within seconds.

In addition to KPIs, you can take advantage of other user-oriented features to help you extend the functionality of a cube. For example, you can implement actions to request URL resources, to drill down into data, and to launch a SQL Server Reporting Services (SSRS) report. You can localize the cube metadata and data to target international users. And you can use perspectives to provide subviews of large and complex cubes.

Understanding KPI *Value*, *Goal, Status*, and *Trend* Properties

In SSAS, a KPI is an extended measure that has four main expression-based properties: *Value*, *Goal*, *Status*, and *Trend*. The *Value* property is required; the rest are optional.

Value

The KPI *Value* property represents the current value of the KPI. This property is typically mapped to a regular or calculated measure, such as Reseller Sales Amount.

Goal

The KPI *Goal* property defines what the KPI value should be in the perfect world. It could be set to a fixed number, such as 0.40 for 40 percent; a regular measure, such as Sales Target; or a calculated measure. The following expression sets the KPI goal to 40 percent more than the Reseller Sales Amount for the previous date period:

```
1.40 * ([Date].[Calendar].PrevMember, [Measures].[Reseller Sales Amount])
```

The *PrevMember* MDX function returns the previous member in a hierarchy. So if the current level is Year, the function returns the previous year; if the current level is Month, you get the previous month; and so on.

Status

The *Status* property indicates how the KPI value compares to the goal. Its expression should return an integer value of –1 for underperformance, 0 for acceptable performance, or 1 for good performance. SSAS provides several standard KPI-related MDX functions: *KPIValue* to retrieve the KPI value, *KPIGoal* to retrieve the KPI goal, *KPIStatus* to retrieve the KPI status, and *KPITrend* to retrieve the KPI trend properties. The following *Status* expression evaluates the range of the KPI value-to-goal ratio and then returns –1, 0, or 1 accordingly:

```
Case
When KpiValue( "Reseller Sales" ) /KpiGoal ( "Reseller Sales" ) >= 1 Then 1
When KpiValue( "Reseller Sales" ) / KpiGoal ( "Reseller Sales" ) < 1
   And KpiValue( "Reseller Sales" ) / KpiGoal ( "Reseller Sales" ) >= .85 Then 0
Else -1
End
```

Trend

The *Trend* property indicates how the KPI value is doing over time. As with the *Status* property, *Trend* should return a value between –1 and 1. For example, the following expression returns the growth ratio in Reseller Sales Amount compared to the previous period:

```
(
   [Measures].[Reseller Sales Amount] –
   ([Date].[Calendar].PrevMember,
   [Measures].[Reseller Sales Amount])
) /
[Measures].[Reseller Sales Amount]
```

Additional KPI Properties

KPIs have some additional metadata properties that could be helpful for a client application, including the following:

- ***Associated Measure Group*** Used to identify the measure group with which this KPI should be associated.
- ***Display Folder*** Used to organize related KPIs into logical folders.
- ***Parent KPI*** Used to indicate the parent KPI of the current KPI. This is useful if a set of KPIs are shown together, as they might be in a scorecard.
- ***Current Time Member*** Used to indicate the member of the Time dimension that should be used as the current member.
- ***Weight*** Used to evaluate the importance of a KPI compared to its siblings.
- ***Status Indicator*** **and** ***Trend Indicator*** Used to identify the images the client should use to display the values graphically.

Creating KPIs

You can use the KPIs tab in the Cube Designer to create and test KPIs. Follow these steps to create a new KPI:

1. Open the cube in the Cube Designer, and then click the KPIs tab, as shown in Figure 6-6.
2. Set the KPI properties.
3. Save your changes, and then deploy your project.

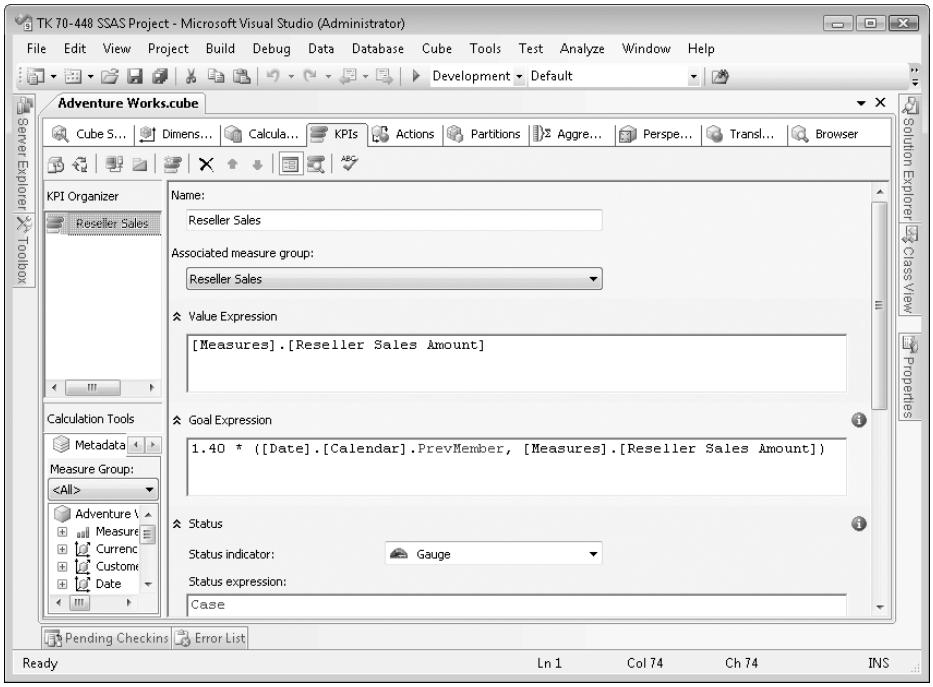


FIGURE 6-6 Use the KPIs tab to create and edit a KPI.

Viewing KPIs

There are a number of ways to test and display KPIs. For quick testing, use the built-in KPI Browser in BIDS, as follows:

1. Deploy the KPI changes to the server.
2. On the KPIs tab in the Cube Designer, select the KPI you want to test in the KPI Organizer pane.
3. Click the Browser View toolbar button to browse the KPIs, as shown in Figure 6-7.
4. Optionally, use the Filter pane of the KPI Browser to filter the KPI properties. For example, to see the Reseller Sales KPI for calendar year 2003, add the Date dimension to the Filter pane, and then set its Calendar hierarchy to 2003.

Although the KPI Browser lets you set up a dimension filter—for example, it will let you filter Calendar Year 2004 only—it does not let you browse the KPIs by multiple dimension members.

You can use KPI-aware client applications, such as Excel 2007 and SSRS, to display KPIs on a report. You can also write an MDX statement in SQL Server Management Studio (SSMS) to directly query the KPI properties by using the *KPIValue()*, *KPIGoal()*, *KPIStatus()*, and *KPITrend()* MDX functions.

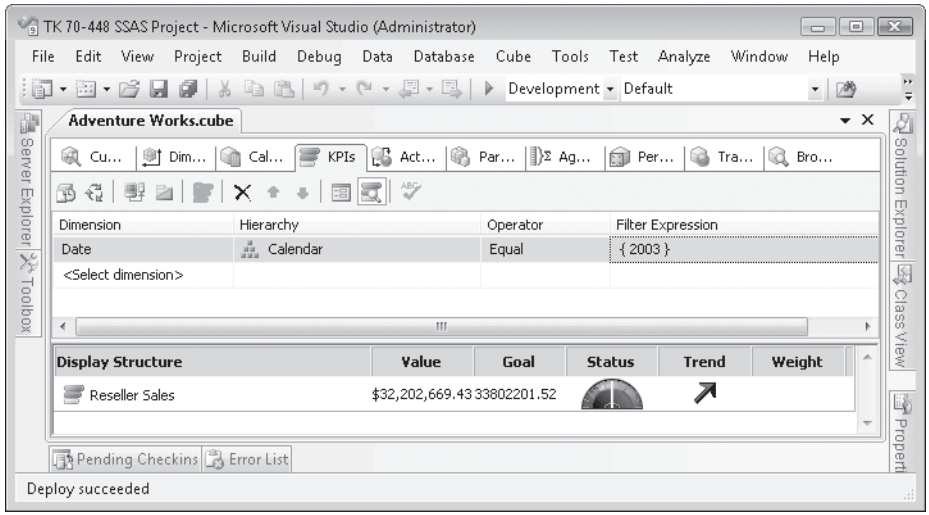


FIGURE 6-7 Use the KPI Browser to test a KPI.

EXAM TIP

You can write an MDX statement that uses the *KPIValue()*, *KPIGoal()*, *KPIStatus()*, and *KPITrend()* MDX functions to query KPIs. For example, the following MDX statement retrieves the properties of the Reseller Sales KPI that you implement in this chapter, sliced by Product Category for the year 2003:

```
SELECT {KPIValue("Reseller Sales"),
KPIGoal("Reseller Sales"),
KPIStatus("Reseller Sales"),
KPITrend("Reseller Sales")} ON COLUMNS,
[Dim Product].[Product Category].Members  ON ROWS
FROM [Adventure Works]
WHERE [Date].[Calendar].[Calendar Year].&[2003]
```

Implementing Actions

Actions can extend your cubes in versatile ways. For example, suppose that a user has drilled down to the lowest level of the Product dimension and wants to see the individual sales orders placed for that product. If the order information is not stored in the cube, you can implement a reporting action that lets the user request an SSRS report that displays the order data from another system.

BEST PRACTICES **UNDERSTAND CLIENT-SPECIFIC FEATURES**

SSAS actions are defined in the cube but are interpreted and initiated by the client application. Not all clients support actions. For example, Excel 2007 supports actions, but SSRS does not. Therefore, before implementing actions, you need to consider the reporting tools your users will use to browse the cube.

Understanding Action Types

You can define several action types to integrate your cube with client applications. The action type informs the client application about how it should interpret the action. There are three main action types that are consistent with the Cube Designer user interface:

- **Regular actions** Multipurpose actions that can retrieve information from different places. Regular actions can be further subdivided based on the action content, as Table 6-4 shows.

TABLE 6-4 Regular Action Types

CONTENT TYPE	PURPOSE
Dataset	The action content is an MDX statement.
Proprietary	The action content is client-specific. The client is responsible for interpreting the semantic meaning of the action.
Rowset	The action content is a command statement to retrieve data. Unlike the Dataset action, however, a Rowset action targets any OLE DB–compliant data source, including a relational database.
Statement	The action content represents an OLE DB command. Unlike the Dataset and Rowset actions, the statement should not yield any results other than success or failure.
URL (default)	The action content is a URL and should indicate one of the standard protocols, such as HTTP, HTTPS, FTP, FILE, or MAIL. For security reasons, a client application might ignore protocols other than HTTP and HTTPS.

- **Drillthrough actions** Let the client request the details behind aggregated cell values in the cube. This is the only action type that the client application can send to SSAS for execution.
- **Reporting actions** Can be used to request SSRS reports. The action command is the URL report path along with optional report parameters.

Creating an Action

Follow these steps to implement a new action:

1. Open the cube in the Cube Designer, and then click the Actions tab, as shown in Figure 6-8.

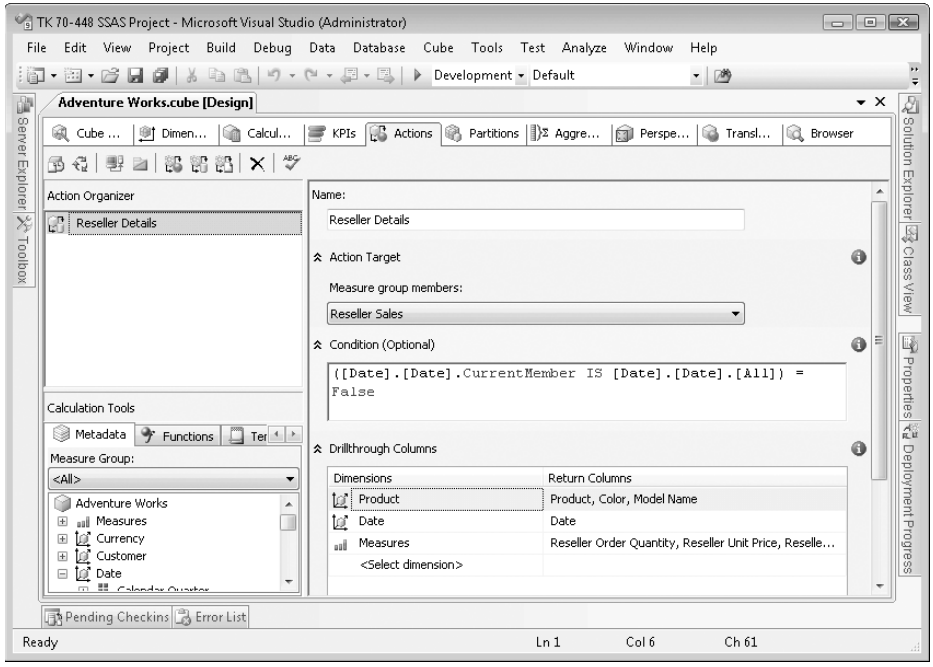


FIGURE 6-8 Use the Actions tab to create an action.

2. Click the New Action, New Drillthrough Action, or New Reporting Action toolbar button to create a regular, drillthrough, or reporting action, respectively.
3. Configure the action properties.

   Common action properties that apply to all action types include action name, target, and condition. Each action type also supports specific properties. For example, when configuring a drillthrough action, you need to specify a list of drillthrough columns that the end user will see when the action is executed.

Understanding Action Discovery

In the process of configuring the action, you specify the action target, which consists of the target type and object. If you want to target an attribute hierarchy, for example, you would choose Attribute members as the target type and the attribute hierarchy as the target object.

Suppose that you have a Web page that takes a product identifier as a query parameter—for example, *http://www.adventureworks.com/olap/ShowTransactions.aspx?ProductName='Road Bike'*—and then displays the order transactions for a given product as the result of that query. Suppose also that the Product dimension has a *ProductName* attribute. For the query to work, you need this action to be available only when the user browses the cube by the *ProductName* attribute hierarchy, so you can get the product name from the *ProductName.CurrentMember.Name* expression. To scope the action this way, set its *Target Type* property to Attribute Members and its *Target Object* property to *Dim Product.ProductName*. Optionally, you can specify an action condition in the form of an MDX expression that further restricts the action scope.

As part of the action discovery stage, the client sends the server the user context, in the form of coordinates, to retrieve the actions defined for that scope. The server matches the action scope against the coordinates to identify the actions that are applicable for that scope. If such actions are discovered, the cube resolves the action command and returns it to the client. The client has final say over whether the action will be presented to and executed by the user.

Localizing Cubes Through Translations

SSAS makes it easy for you to support international users by defining translations in the cube. As its name suggests, a *translation* is a translated version of cube metadata (such as captions of measures, dimensions, perspectives, and KPIs) and data (members of attribute hierarchies).

Translating Cubes

To translate the cube metadata—including captions of cubes, cube dimensions, measure groups, measures, KPIs, actions, and named sets—open the Cube Designer, and then click the Translations tab. Click the New Translation button on the toolbar, and then select the language for which you want to define a translation. For each element in the cube, enter a caption that has been *localized* (translated to a given foreign language).

Translating Dimensions

Follow these steps to implement a dimension translation:

1. Open the dimension in the Dimension Designer, and then click the Translations tab.
2. Click the New Translation button on the toolbar, and from the drop-down list, select the language for which you want to define a translation. Enter a localized caption for each element in the cube.

3. To localize dimension members, select the intersecting cell between the attribute hierarchy and the language column, and then click the ellipsis (…) button.
4. In the Attribute Data Translation dialog box that appears, select the column of the dimension table that stores the translated caption, as shown in Figure 6-9.

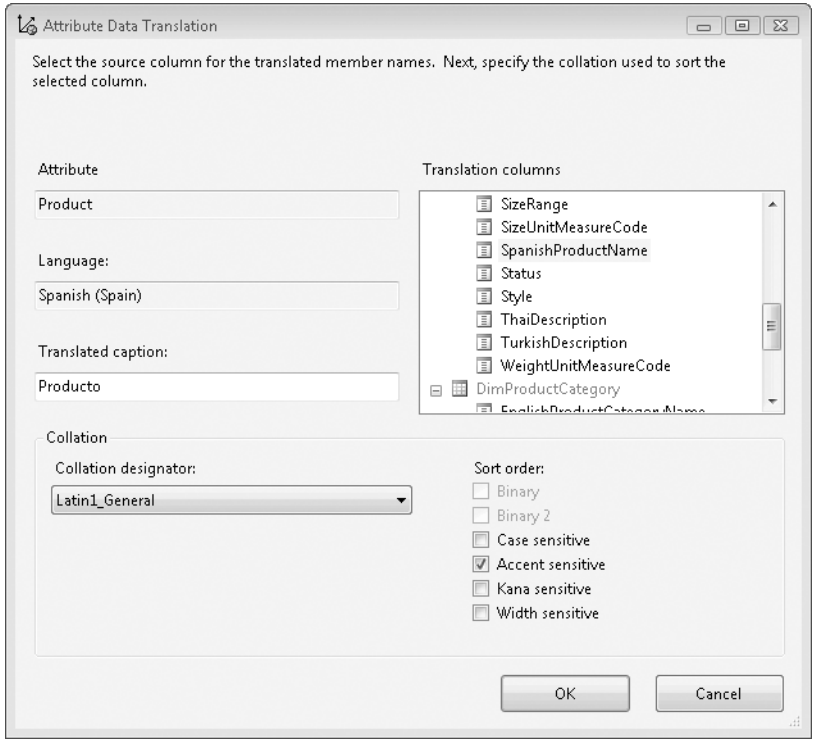


FIGURE 6-9 Use the Attribute Data Translation dialog box to localize the attribute caption and data.

By default, SSAS selects a translation based on the local culture of the current thread. At design time, the easiest way to test translations is to use the Cube Browser and select the desired language from the Languages drop-down list.

Implementing Cube Perspectives

SSAS cubes can become quite large and span several measure groups, but a large dimensional model might be overwhelming to end users. This is where perspectives can help.

A *cube perspective* represents a subset of the cube. Its main purpose is to reduce the perceived complexity of a large cube by exposing only a subset of the cube objects. For example, if the Adventure Works reseller sales department is mostly interested in browsing reseller sales, you can create a perspective that includes only the Reseller Sales measure group and its associated dimensions. By default, the cube has a single perspective that exposes the entire cube content.

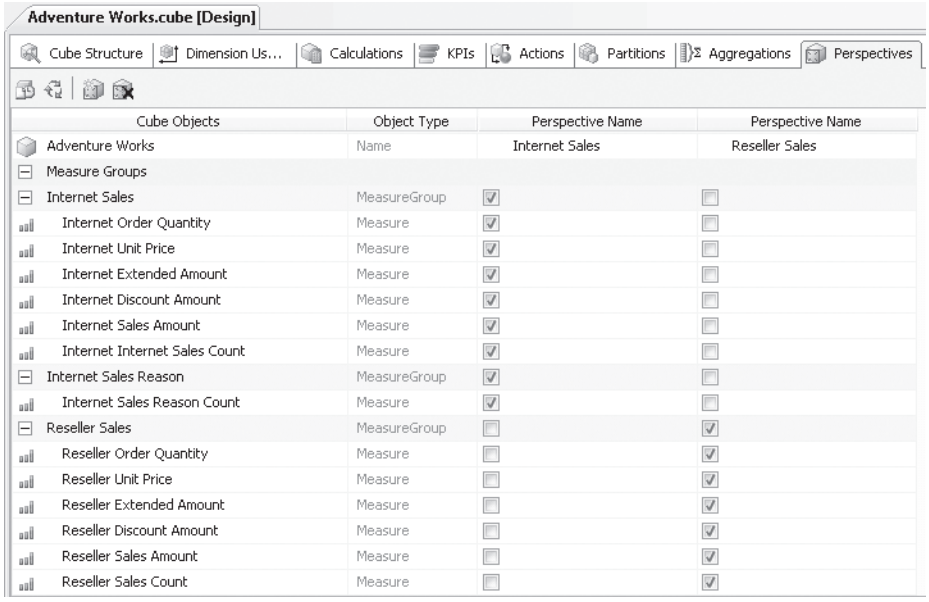
IMPORTANT **PERSPECTIVES ARE NOT A SECURITY MECHANISM**

Perspectives are not a security mechanism, and you cannot use them to enforce restricted access to portions of the cube. Object security policies pass through the containing perspectives. For example, if the user does not have access to a given dimension, that dimension will not show in the perspectives that contain it.

Defining Perspectives

You can create new perspectives by using the Perspectives tab of the Cube Designer. To do so, follow these steps:

1. Open the cube in the Cube Designer, and then click the Perspectives tab.
2. Click the New Perspective button on the toolbar, and then specify the name of the perspective.
3. Select the objects to be included in the perspective. These objects can include measure groups, measures, dimensions, hierarchies, attributes, KPIs, actions, and calculations. For example, Figure 6-10 shows a cube that has Internet Sales and Reseller Sales perspectives to provide logical views pertinent to the Internet and reseller sales subject areas.



Cube Objects	Object Type	Perspective Name	Perspective Name
Adventure Works	Name	Internet Sales	Reseller Sales
Measure Groups			
Internet Sales	MeasureGroup	☑	☐
Internet Order Quantity	Measure	☑	☐
Internet Unit Price	Measure	☑	☐
Internet Extended Amount	Measure	☑	☐
Internet Discount Amount	Measure	☑	☐
Internet Sales Amount	Measure	☑	☐
Internet Internet Sales Count	Measure	☑	☐
Internet Sales Reason	MeasureGroup	☑	☐
Internet Sales Reason Count	Measure	☑	☐
Reseller Sales	MeasureGroup	☐	☑
Reseller Order Quantity	Measure	☐	☑
Reseller Unit Price	Measure	☐	☑
Reseller Extended Amount	Measure	☐	☑
Reseller Discount Amount	Measure	☐	☑
Reseller Sales Amount	Measure	☐	☑
Reseller Sales Count	Measure	☐	☑

FIGURE 6-10 Implement perspectives to define logical views of the cube metadata.

PRACTICE Creating KPIs, Actions, Translations, and Perspectives

In this practice, you will see how to create KPIs, actions, translations, and perspectives to help Adventures Works employees track reseller performance, drill through data, view cube information in Spanish, and see only the data they need to see.

EXERCISE 1 Implement the Reseller Sales KPI

In this exercise, you will learn how to implement a Reseller Sales KPI for Adventure Works to track the reseller sales against a predefined goal.

1. Open the TK 70-448 SSAS Project in BIDS (the ..\Source\Ch 06\TK448-ch06 Lesson 1\ folder in the installed practice files within the Documents folder for your user account).
2. Open the Adventure Works cube in the Cube Designer, and then click the KPIs tab.
3. Click the New KPI button on the toolbar.
4. Type **Reseller Sales** in the Name box.
5. Select the Reseller Sales measure group in the Associated Measure Group drop-down list.
6. For the sake of simplicity, the value of the Reseller Sales KPI will be supplied from the existing Reseller Sales Amount measure. Type **[Measures].[Reseller Sales Amount]** in the Value Expression box.
7. For the current period, Adventure Works wants to see reseller sales increase 40 percent over the previous period. Enter the following expression in the Goal Expression box:

   ```
   1.40 * ([Date].[Calendar].PrevMember, [Measures].[Reseller Sales Amount])
   ```

   Alternatively, you can select the Metadata and Functions tabs to drag metadata objects and MDX functions from the Calculations Tools pane to construct the expression. Note that the KPI Designer uses color coding in the expression to emphasize reserved words and functions.
8. Leave Gauge as the status indicator. Enter the following expression in the Status box:

   ```
   Case
   When KpiValue("Reseller Sales")/KpiGoal("Reseller Sales") >= 1
   Then 1
   When KpiValue("Reseller Sales")/KpiGoal("Reseller Sales")< 1
   And KpiValue("Reseller Sales")/KpiGoal("Reseller Sales")>= .85
   Then 0
   Else -1
   End
   ```

9. Leave the Trend Indicator drop-down list set to Standard Arrow. Enter the following expression in the Trend Expression box:

```
([Measures].[Reseller Sales Amount] -
([Order Date].[Calendar].PrevMember,
[Measures].[Reseller Sales Amount]))/
[Measures].[Reseller Sales Amount]
```

10. Deploy the solution to send the changes to the server and process the Adventure Works cube.
11. Now that the Reseller Sales KPI is ready, you will test it in the Browser view of the KPI Designer. With the KPIs tab still open, click the Browser View button on the toolbar, and then click the Reconnect button to create a new session.
12. In the Filter pane, expand the Dimension drop-down list, and then select the Date dimension. Expand the Hierarchy drop-down list, and then select the Calendar user hierarchy. Expand the Filter Expression drop-down list, and then select the year 2003 check box and click OK. The values of the Reseller Sales KPI change to reflect the filter selection.

EXERCISE 2 Implement a Drillthrough Action, a Translation, and Perspectives

In this exercise, you will learn how to implement a drillthrough action, a dimension translation, and two perspectives. Suppose that the end users want to see the individual resale order transactions behind a given aggregate cell in the cube. Follow these steps to implement the Reseller Details drillthrough action:

1. Open the Adventure Works cube in the Cube Designer.
2. Click the Actions tab.
3. Click the New Drillthrough Action button on the toolbar. The Cube Designer creates an empty action named Drillthrough Action.
4. Type **Reseller Details** in the Name box.
5. A drillthrough action can be associated with measure groups only. Expand the Measure Group Members drop-down list, and then select Reseller Sales.
6. A drillthrough action can potentially return many rows. You can use an MDX condition to limit the scope of the action. Suppose that you want the Reseller Sales drillthrough action to be activated only when the user drills down to the Date level of the Date dimension. To achieve this, enter the following expression in the Condition box:

```
([Date].[Date].CurrentMember IS [Date].[Date].[All]) = False
```

7. Use the Drillthrough Columns pane to specify the columns to show to the end user when the user initiates the drillthrough action. To do so, from the Dimensions drop-down list, select a dimension such as Product. Then, in the Return Columns drop-down list, select the check box for the columns you want to display. You can choose any measures from the targeted measure group and/or attributes from the dimension joined to it. Select the Product, Color, and Model Name attribute check boxes of the Product dimension, the Date attribute of the Date dimension, and all the measures that are part of the Reseller Sales measure group.
8. Expand the Additional Properties pane. To be sure that the action does not return too many rows and cause performance issues, type **100** in the Maximum Rows box.
9. Deploy the project. To test the drillthrough action, navigate to the Browser tab, and then create a pivot report with the Calendar hierarchy of the Date dimension on columns and the Reseller Sales Amount measure on data.
10. Assuming that the Calendar hierarchy is not expanded to its lowest level, right-click a Reseller Amount cell and notice that the Reseller Details action is not shown.
11. Expand the Calendar hierarchy to the Date level. Again, right-click a Reseller Amount cell, and notice that the Reseller Details action is now available, as Figure 6-11 shows.

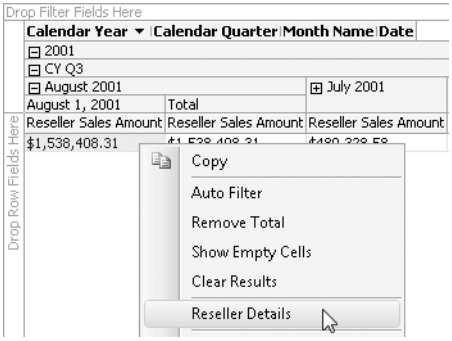


FIGURE 6-11 The client application is responsible for showing available actions for user selection.

12. Click Reseller Details to initiate the action, and notice that a new dialog box named Data Sample Viewer opens. The Data Sample Viewer dialog box shows the actual transactions that were loaded from the fact table when the cube was processed.

EXERCISE 3 Implement a Dimension Translation

Suppose that the Adventure Works cube will be browsed by Spanish-speaking users. Follow these steps to localize the Product dimension for Spanish speakers by implementing a dimension translation:

1. In Solution Explorer, double-click the Product dimension to open it in the Dimension Designer, and then click the Translations tab.
2. Click the New Translation button on the toolbar, select the Spanish (Spain) language in the Select Language dialog box that appears, and then click OK. A new column titled Spanish (Spain) is added to the grid.
3. Localize the dimension name to Spanish by typing **Producto** in the Dim Product row. Localize the name of the Product attribute hierarchy, the dimension key, by typing **Producto** in the Product row.
4. To localize the captions of the members in the Product hierarchy, select the intersecting cell between the Spanish (Spain) column and the Product attribute hierarchy. Click the ellipsis (…) button inside the cell to open the Attribute Data Translation dialog box.
5. In the Translation Columns list, select the SpanishProductName column, and then click OK. Notice that the Producto cell now has an icon that indicates that the attribute data for this attribute hierarchy has been localized.
6. To test the Spanish translation, deploy the solution, and then open the Adventure Works cube in the Cube Designer. Click the Browser tab. Create a report that uses the Product attribute hierarchy on rows and Internet Sales Amount on data.
7. Expand the Language drop-down list, and then select Spanish (Spain). Notice that the captions of the Product attribute hierarchy and its members are now in Spanish, as Figure 6-12 shows.

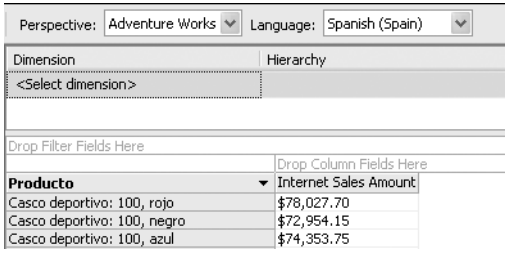


FIGURE 6-12 To test translations in the Cube Browser, set the Language drop-down list to the required language.

EXERCISE 4 Implement a Perspective

As the Adventure Works cube grows in complexity, users might find it difficult to navigate through its metadata. In this exercise, you will create Internet Sales and Reseller Sales perspectives to show only selected objects of the cube.

1. Open the Adventure Works cube in the Cube Designer, and then navigate to the Perspectives tab.
2. Click the New Perspective button on the toolbar. A new column is added to the grid.
3. Change the name of the new perspective from Perspective to **Internet Sales**.
4. Clear the check boxes for the Reseller Sales measure group, the Reseller Sales KPI, and the Reseller Details action.
5. Repeat steps 2 and 3 to create a new **Reseller Sales** perspective.
6. Clear the check boxes for the Internet Sales, Internet Sales Reason, and Internet Customers measure groups and the Customer, Sales Reason, and Internet Sales Order Details dimensions.
7. Deploy the solution, and then navigate to the Browser tab in the Cube Designer.
8. Expand the Perspective drop-down list, and then select the Reseller Sales perspective. Notice that the cube metadata pane changes to show only the objects that are included in the Reseller Sales perspective, as Figure 6-13 shows.

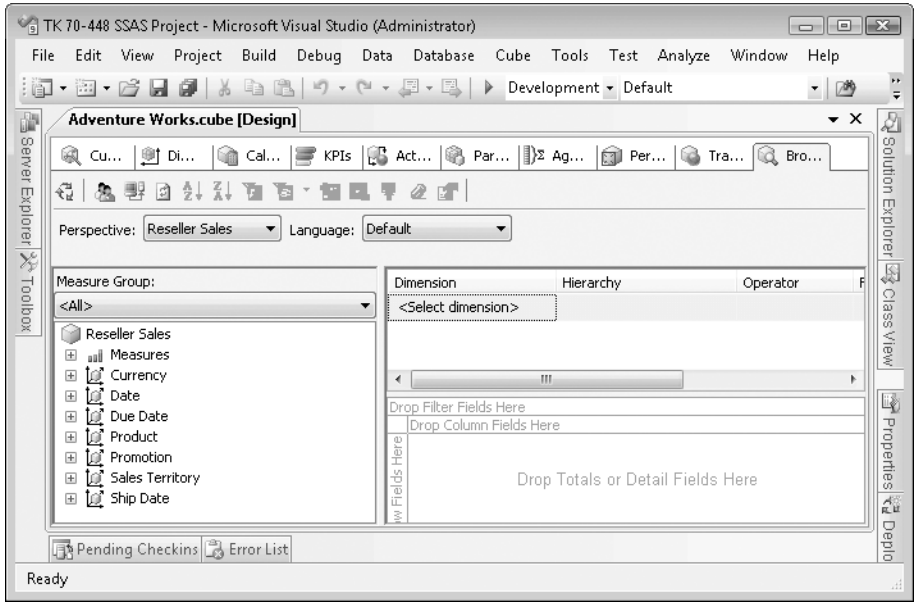


FIGURE 6-13 The Reseller Sales perspective filters the cube metadata to show only the cube objects that are included in the perspective.

Quick Check

1. What types of actions can you identify?
2. How you can localize dimension members' captions?
3. Can you use perspectives to enforce security?
4. What is a KPI?
5. What are the main properties of a KPI in SSAS?
6. What will the KPI Status expression return to indicate underperformance?

Quick Check Answers

1. Regular, drillthrough, and reporting actions are the three types of actions available.
2. You can localize dimension members' captions by selecting a translation column that stores the translated captions.
3. No, you can use perspectives to make the cube easier to browse but not as a security mechanism.
4. A KPI, or key performance indicator, is a quantifiable measure used to track business performance.
5. The main properties of an SSAS KPI are *Value*, *Goal*, *Status*, and *Trend*.
6. The KPI Status expression will return –1 to indicate underperformance.

Lesson 3: Creating Calculations and Queries by Using MDX

Estimated lesson time: 45 minutes

SSAS lets you quickly build dimensional models that provide essential OLAP and data mining features. However, chances are that in real life, business needs will require you to go beyond what the dimensional model can provide and enhance the cube with business logic. MDX gives you the programming power to implement custom calculations and unlock the full potential of SSAS. This lesson provides essential coverage of MDX programming by explaining how you can work with MDX queries, calculated members, and named sets.

Understanding MDX Syntax

Although originally developed for SSAS, over the years MDX has become the common language of OLAP. Most OLAP vendors have embraced or are currently adopting the XML for Analysis (XMLA) specification (see References), which describes the MDX language. As a result,

there are many OLAP servers and browsers on the market that use MDX as a programming language in one form or another.

In SSAS, MDX is used in two main ways: to query and to extend multidimensional cubes. Client applications can send MDX queries to the cube to retrieve results. You can also use MDX expressions to extend your cubes. For example, you can use MDX to implement business metrics such as KPIs.

MDX Fundamentals

The results of an MDX query or expression depend on the current context in the cube. To use MDX effectively, you need to understand how to navigate the cube space by referencing tuples and sets.

TUPLES

A *tuple* is a multidimensional coordinate that identifies a single cell in the cube space. A tuple is produced by one member that is taken from one or more attribute hierarchies. For example, the tuple [Product].[Product].[Road Bike], [Date].[Year].[2004], [Measures].[Internet Sales] references a cell found at the intersection of product Road Bike, year 2004, and measure Internet Sales; the cube measures are treated as members of a special Measures dimension.

> ***BEST PRACTICES*** **REFERENCE DIMENSION MEMBERS**
>
> **When you reference a dimension member in a tuple, you can use the member name (for example, [Product].[Product].[Mountain-100]). However, this syntax works only if there is a single product with that name. Rather than using the member name, you can resolve the member uniquely by using its key. To reference a member by key, you prefix the key with an ampersand (&). Assuming that you have a product with a key of 10, [Product].[Product].&[10] will resolve that member uniquely, even if there are other products with the same name.**

Because the cube space typically consists of many attribute hierarchies, enumerating each of them is tedious. When you omit an attribute hierarchy, its default member (usually the hierarchy *All* member) is used.

SETS

An *MDX set* is a collection of tuples with the same dimensionality, or attribute hierarchies. For example, the following set is a valid set consisting of two tuples that have the same dimensionality:

```
{([Sales Territory].[Country].[All Countries],[Date].[Year].[2004], [Product].[Product].
[Mountain-100]),
([Sales Territory].[Country].[All Countries],[Date].[Year].[2004], [Product].[Product].
[Road-200])}
```

The following set is invalid because its tuples have different dimensionality: The first tuple uses the Customer dimension, and the second uses the Territory dimension:

```
{([Customer].[John Doe], [Date].[Year].[2003], [Product].[Product].[Mountain-100]),
([Sales Territory].[Country].[Canada],
[Date].[Year].[2003], [Product].[Product].[Road-200])}
```

Basic MDX Queries

The basic MDX query consists of a *SELECT* statement that has the following syntax:

```
SELECT [ * | ( <SELECT query axis clause>
    [ , <SELECT query axis clause> ... ] ) ]
FROM <SELECT subcube clause>
[ <SELECT slicer axis clause> ]
```

For example, the following MDX query returns the Internet Sales Amount measure on columns, broken down by product categories or rows, and filtered for calendar year 2004 only:

```
Select [Measures].[Internet Sales Amount] on Columns,
    [Product].[Category].Members on Rows
From [Adventure Works]
Where [Date].[Calendar Year].&[2004]
```

QUERY AXES

A query can have up to 128 axes (numbered from 0 to 127). The preceding query uses only two. The first five axes are named COLUMNS, ROWS, PAGES, SECTIONS, and CHAPTERS. Note that although you can project the results on many axes, you cannot skip axes. For example, you cannot request the ROWS axis without the COLUMNS axis. Rather than using the axis alias, you can use the ordinal position for the axis—for example, you can use 0 in place of COLUMNS.

Typically, the *FROM* clause specifies the name of the cube to query. Alternatively, the *FROM* clause can use another *SELECT* statement that defines a portion, or subcube, of the cube. The slicer axis filters the query results.

CALCULATED MEMBERS

Optionally, the query can request calculated columns in the form of calculated members by using a *WITH* clause before the *SELECT* statement. For example, the following query defines a *Gross Profit* calculated member as a sum between the Internet Sales Amount and Reseller Sales Amount measures:

```
With  Member [Gross Profit] AS
   '[Measures].[Internet Sales Amount] +
    [Measures].[Reseller Sales Amount]'
Select {[Measures].[Internet Sales Amount],[Gross Profit]}
on Columns,
      [Product].[Category].Members on Rows
```

```
From [Adventure Works]
Where [Date].[Calendar Year].&[2003]
```

Frequently used calculated members can be defined inside the cube so that they are readily available to all clients.

Applying MDX Functions

MDX provides a host of useful functions to navigate the cube space. Here are a few of the most commonly used functions.

The *CurrentMember* Function

The *CurrentMember* function takes a hierarchy and returns the current member in respect to a given cell coordinate. For example, the tuple [Order Date].[Calendar].CurrentMember, [Measures].[Reseller Sales Amount] returns the Reseller Sales Amount for the member of the Calendar hierarchy that references the current cell. The *CurrentMember* is the default property of a dimension member and often is omitted.

Functions for Navigating Hierarchies

MDX provides functions such as *PrevMember*, *Children*, and *Parent* for navigating hierarchies. *PrevMember* takes a member and returns a previous member in the hierarchy. For example, [Order Date].[Calendar].PrevMember returns the previous member with respect to the current member of the Calendar hierarchy. So if the Calendar hierarchy is expanded to the Year level and the current cell is referenced by year 2004, *PrevMember* will return year 2003.

EXAM TIP

The *PrevMember* function is frequently used to define the KPI Trend or Goal properties. For example, the Reseller Sales KPI uses the following expression to set the Goal property to 40 percent more than the Reseller Sales Amount for the previous date period:

```
1.40 * ([Date].[Calendar].PrevMember, [Measures].[Reseller Sales Amount])
```

Consequently, if the user browses the cube data by years, the *PrevMember* function will return the previous year for each year. If the user browses data by quarters, the *PrevMember* will return the previous quarter for each quarter, and so on.

The *Members* function returns all members of a hierarchy. For example, [Product].[Product].[Product].Members returns all products excluding the *All* member. The *Children* function returns the members below a given member. For example, [Date].[Calendar].[Calendar Year].[2003].Children returns all quarters of year 2003, assuming that the level below the Year level in the Calendar hierarchy is Quarter. In contrast, the *Parent* function returns the member's parent in a given hierarchy. So [Product].[Categories].[Product]. [Road Bike].Parent returns the *Bikes* member of the Product Category hierarchy.

Functions for Navigating Time

MDX provides a few functions, such as *Lag* and *ParallelPeriod*, to help you navigate the Time dimension. The *Lag* function returns a member at a given offset from a given member—for example, [Order Date].[Calendar].Lag(1) returns the previous sibling of the current member. So if the current member is at the Quarter level, *Lag(1)* will return the previous quarter. If it is at the Year level, *Lag(1)* will return the previous year, and so on.

ParallelPeriod returns the member from a prior period in the same relative position as a specified member with an optional offset; if no member is specified, 0 is assumed. For example, ParallelPeriod ([Date].[Calendar].[Calendar Quarter], [Date].[Calendar].[Month Name].[October 2003]) returns October 2002.

EXAM TIP

Another time-related function commonly used in MDX expressions is the *PeriodsToDate* function. This function returns a set of sibling members from the same level as a given member, starting with the first sibling and ending with the given member. For example, the following query uses the *PeriodsToDate* function to return the Internet sales for the first 10 months of 2003:

```
select [Measures].[Internet Sales Amount] on 0,
PeriodsToDate([Date].[Calendar].[Calendar Year],
[Date].[Calendar].[Month Name].&[10]&[2003]) on 1
from [Adventure Works]
```

For a list of all MDX functions and how to use them, see the topic "Multidimensional Expressions (MDX) Reference" in SQL Server 2008 Books Online, included in the References section at the end of this book.

Creating Calculated Members

A calculated member is a dimension member whose value is calculated dynamically at run time. Remember that the cube measures are considered members of a special Measures dimension. The definitions of the cube calculated members and named sets become part of the cube MDX script, which can be accessed from the Calculations tab of the Cube Designer. Unlike a regular measure, a calculated member does not increase the cube size because the server stores only its definition, not its data.

Calculated Member Syntax

The following statement creates a *Sales Amount* calculated member as the sum between the Internet Sales Amount and Reseller Sales Amount measures:

```
CREATE MEMBER CURRENTCUBE.[MEASURES].[Sales Amount]
 AS [Measures].[Internet Sales Amount] +
   [Measures].[Reseller Sales Amount],
FORMAT_STRING = "Currency",
VISIBLE = 1;
```

You define a calculated member by using the *CREATE MEMBER* MDX statement, which uses an MDX expression. A calculated member has a few properties such as *FORMAT_STRING* and *Visible* that control its display behavior.

When to Use Calculated Members

As you investigate whether to implement a calculated member, you have the following options:

- If you need a measure that does not belong to a particular measure group, consider defining it as a calculated member. For example, the Sales Amount measure spans both Internet Sales and Reseller Sales measure groups. Thus, it is a good candidate to become a calculated member.
- Consider creating a calculated member when you need a nonadditive measure that operates uniformly across dimension levels. For example, a calculated member that calculates the average reseller profit amount (Reseller Sales Amount / Reseller Sales Count) returns the expected results no matter how the end user slices the cube. It simply takes the measure values at the current cube coordinate and divides them. In comparison, an additive measure with a *SUM* aggregate function will produce incorrect results because it will sum up the values.

> ***NOTE*** **SCOPE ASSIGNMENTS**
>
> **Scope assignments let developers write to the cube space and overwrite the measure aggregation behavior. In general, however, calculated members do not have a default aggregation function. If you need to scope the expression at a certain level of a dimension—at the leaf member level, for example—and let the server aggregate the measure from that level up, it is probably better to use standard measure and scope assignments. Scope assignments are not covered in this book.**

Defining Named Sets

As its name suggests, a named set is an MDX construct that has an alias and that returns a set of dimension members. You can explicitly specify the set tuples, or you can use standard set-producing MDX functions, such as *Children* or *Members*. For example, the following named set returns the top 50 customers:

```
CREATE SET CURRENTCUBE.[Top 50 Most Profitable Customers]
 AS
    TopCount (
      (Existing [Customer].[Customer].[Customer].Members),
                 50,
                [Measures].[Internet Sales Amount]
          );
```

The *Members* function returns all members of the Customer attribute hierarchy in the Dim Customer dimension. Use the triple notation syntax, *Dimension.Attribute.Attribute*, to exclude the *All* member of the hierarchy. The *TopCount* function ranks the set by evaluating the customer *[Internet Sales Amount]* total and returns the top 50 customers only. Last, the *Existing* keyword forces the set to be reevaluated within the current context, such as when the user selects a different time period.

SSAS supports two types of named sets based on how the server evaluates them. A *static named set* is evaluated once by the server. In contrast, a *dynamic named set*, which was introduced in SSAS 2008, is evaluated for each query. By default, BIDS creates dynamic named sets. For more information about static and dynamic named sets, see the blog entry "MDX in Katmai: Dynamic Named Sets" by Mosha Pasumansky (see References).

PRACTICE Extending Cubes by Using MDX Expressions

In this practice, you will create MDX queries, calculated members, and named sets.

EXERCISE 1 Create and Execute MDX Queries

In this exercise, you will learn how to use SSMS to create and test an MDX query.

1. Open SSMS.
2. Press Ctrl+Alt+T to open Template Explorer.
3. In Template Explorer, click the Analysis Server button on the toolbar to see the Analysis Services templates only.
4. Expand the MDX node, expand the Queries node, and then double-click the Basic Query template. SSMS should generate the following query:

```
Select <row_axis, mdx_set,> on Columns,
     <column_axis, mdx_set,> on Rows
From <from_clause, mdx_name,>
Where <where_clause, mdx_set,>
```

5. When prompted, connect to the SSAS server.
6. Select the TK 70-448 SSAS Project database in the Available Databases drop-down list. SSMS shows the metadata of the Adventure Works cube in the Cube pane.
7. In the Cube pane, expand the Measures folder and the Internet Sales measure group, and then drag the Internet Sales Amount measure before the ON Columns clause of the query. Delete the <row_axis, mdx_set,> clause.
8. In the Cube pane, expand the Product dimension. Drag the Category attribute hierarchy onto the query ROWS axis. Type **.Members** after [Product].[Category] so that it becomes [Product].[Category].Members. Delete the <column_axis, mdx_set,> clause.
9. In the Cube pane, drag the root node, Adventure Works, so that it follows the FROM keyword. Delete the <from_clause, mdx_name,> clause.

10. Expand the Date dimension, expand Date.Calendar Year and Calendar Year, and then drag 2003 so that it follows the WHERE clause of the query. Delete <where_clause, mdx_set,>.
11. Click the Execute button on the toolbar to execute the query and see the results. Your screen should match the screen shown in Figure 6-14.

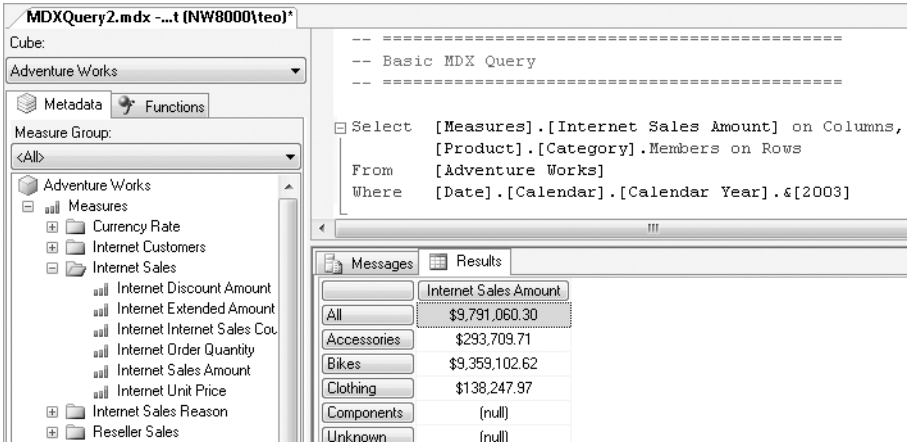


FIGURE 6-14 Use SSMS to write and test MDX queries.

EXERCISE 2 Implement a Calculated Member

In this exercise, you will use BIDS to implement a *Sales Amount* calculated member whose definition is saved in the cube script. You will define the *Sales Amount* calculated member as a sum of the Internet Sales Amount and Resellers Sales Amount measures.

1. Open the TK 70-448 SSAS Project in BIDS (the ..\Source\Ch 06\TK448-ch06 Lesson 2 folder of the installed practice files).
2. Open the Adventure Works cube in the Cube Designer, and then navigate to the Calculations tab. By default, the Form View button on the toolbar is selected, which lets you work with one script object at a time.
3. Click the New Calculated Member button on the toolbar.
4. Type **[Sales Amount]** as the calculated member name.
5. Enter the following expression in the Expression box:

```
[Measures].[Internet Sales Amount] +
[Measures].[Reseller Sales Amount]
```

6. Select Currency from the Format String drop-down list, as Figure 6-15 shows.
7. Click the Calculation Properties button on the toolbar.
8. In the Calculation Properties dialog box, expand the Calculation Name drop-down list, and then select the [Measures].[Sales Amount] calculated member.

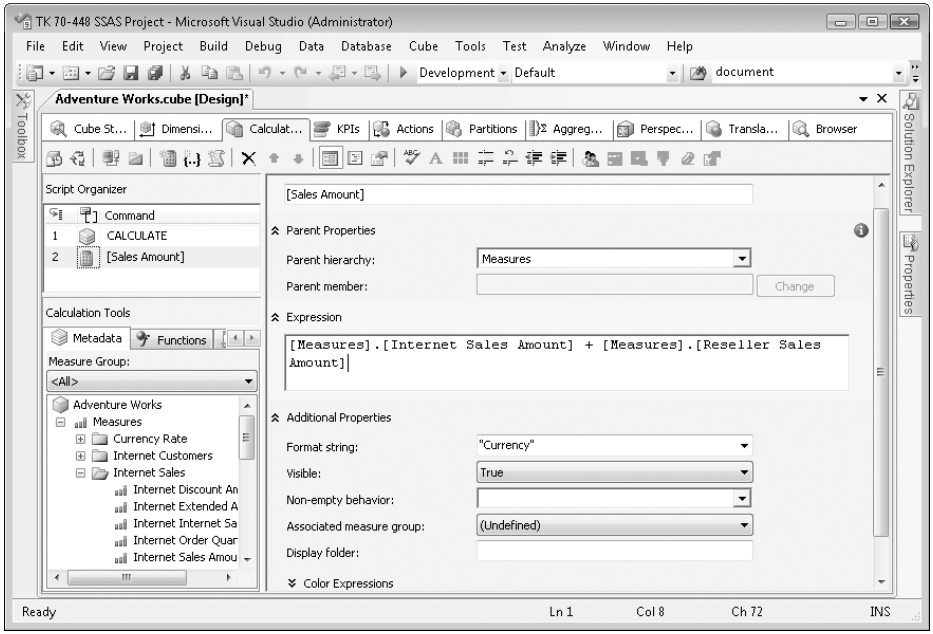


FIGURE 6-15 Use the Form View toolbar of the Calculations tab to enter the properties of the calculation member in predefined fields.

9. In the Display Folder column, type **Sales Summary**. Leave the Associated Measure Group column empty—because the Sales Amount calculated member spans two measure groups, it cannot be logically associated with either of them. Click OK to close the Calculation Properties dialog box.
10. Deploy the solution, and then click the Browser tab.
11. In the Measure Group pane, expand the Measures node, and notice that there is a new Sales Summary folder.
12. Expand the Sales Summary folder. Notice that the Sales Amount calculated member has a special icon to differentiate it from the regular measures.
13. To test the Sales Amount calculated member, drag it to the data section of the report.

EXERCISE 3 Implement a Named Set

In this exercise, you will implement a named set in BIDS that will return the top 50 customers by evaluating the Internet Sales Amount measure.

1. With the Adventure Works cube open in the Cube Designer, navigate to the Calculations tab.
2. Click the New Named Set button on the toolbar.

3. In the Name box, type **[Top 50 Most Profitable Customers]**.
4. Enter the following expression in the Expression box:

```
TopCount (
(EXISTING [Customer].[Customer].[Customer].Members),
50, [Measures].[Internet Sales Amount])
```

5. Deploy the solution, and then switch to the Browser tab. If the report pane has existing results displayed, select the results and then click the Clear Results button on the toolbar to clear them.

   The Browser does not support named sets on reports. However, you can test a named set as a filter to the report.

BEST PRACTICES **USE A PIVOTTABLE REPORT TO TEST NAMED SETS**

You can create a PivotTable report in Excel 2007 to test named sets on report columns or rows.

6. Expand the Customer dimension. SSAS has discovered that the [Top 50 Most Profitable Customers] named set uses members of the Customer dimension, so it has automatically associated the named set with the Customer dimension. Although you cannot assign the named set to another dimension or measure group, you can use the Calculation Properties window to assign the named set to a display folder inside the Customer dimension.
7. Drag the [Top 50 Most Profitable Customers] named set onto the dimension filter.
8. Drag the Customer attribute hierarchy onto rows and Internet Sales Amount onto data, as Figure 6-16 shows. Notice that only the top 50 customers are returned.

Dimension	Hierarchy	Operator	Filter Expression
Customer	Customer	In	Top 50 Most Profitable Customers
<Select dimension>			

Drop Filter Fields Here

Drop Column Fields Here

Customer	Internet Sales Amount
Aaron Wright	$10,813.63
Adriana Gonzalez	$13,242.70
Alicia Shen	$9,458.18
Amy Sun	$9,780.04
Andres Nara	$10,789.53
Anne Dominguez	$8,886.53

FIGURE 6-16 Test the named set in the Browser tab of Cube Designer by dragging it onto the dimension filter.

Quick Check

1. What are the first two axes of an MDX query?
2. What is the difference between a calculated member and a regular measure in terms of storage?
3. What is a named set?

Quick Check Answers

1. COLUMNS and ROWS are the first two axes of an MDX query.
2. The values of a regular measure are stored on the disk, and the values of a calculated member are calculated at run time.
3. A named set is a set consisting of dimension members.

Case Scenario: Extending SSAS Cubes

As it stands, the Adventure Works cube has data only for Internet sales. However, the business requirements state that the reporting solution must support consolidated reports that show both Internet and reseller sales. Adventure Works is also in the process of developing a Web-based dashboard solution, which needs to display vital business metrics in the form of KPIs.

1. What do you need to change in the dimensional model to accommodate the reseller sales?
2. How could you implement the Adventure Works KPIs?

Chapter Summary

- The cube space is defined by attributes. Sometimes there are logical relationships among attributes within the same dimension. You should understand and explicitly define such relationships to optimize the UDM.
- In addition to attribute hierarchies, you can define user hierarchies that provide useful navigation paths in the UDM. A dimension can have several user hierarchies.
- You can browse data in a measure group by a dimension only if the dimension is joined to the measure group. You must review the Dimension Usage tab and correct the dimension relationships if needed. In addition to regular relationships, SSAS supports other relationship types to support more complex schemas.
- Key performance indicators (KPIs) are quantifiable measures that organizations can use to track business performance. A KPI has four main properties: *Value*, *Goal*, *Status*, and *Trend*.

- You can use cube actions to extend your cube functionality. With the exception of the drillthrough action, which is handled by SSAS, all other actions must be handled by external applications.
- Translations let modelers localize cube metadata, dimension metadata, and dimension member captions to foreign languages to support international users.
- Perspectives expose subviews of the cube metadata to reduce the perceived complexity of the cube.
- You can implement business calculations in the form of MDX calculated members and named sets.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

About the Authors

ERIK VEERMAN (SQL Server MVP) is a mentor for Solid Quality Mentors focusing on training, mentoring, and architecting solutions on the Microsoft SQL Server business intelligence (BI) platform. His industry recognition includes the Microsoft Worldwide BI Solution of the Year and *SQL Server Magazine* Innovator Cup. Erik has designed dozens of BI solutions across a broad business spectrum—telecommunications, marketing, retail, commercial real estate, finance, supply chain, and information technology. As an expert in OLAP design, ETL processing, and dimensional modeling, Erik is a presenter, an author, and an instructor. He led the ETL architecture and design for the first production implementation of SQL Server Integration Services (SSIS) and helped drive the ETL standards and best practices for SSIS on the Microsoft SQL Server 2005 reference initiative, Project REAL. Erik is coauthor of the four Wrox books that focus on SQL Server Integration Services and was also the lead author for *MCTS Self-Paced Training Kit (Exam 70-445): Microsoft SQL Server 2005 Business Intelligence—Implementation and Maintenance* (Microsoft Press, ISBN 978-0-7356-9156-8).

TEO LACHEV (MCSD, MCT) works as a technical architect for a leading financial institution, where he designs and implements Microsoft-centric BI solutions. A Microsoft SQL Server MVP since 2004, Teo is also the author of *Applied Microsoft SQL Server 2008 Reporting Services* and *Applied Microsoft Analysis Services 2005*, and the coauthor of *MCTS Self-Paced Training Kit (Exam 70-445): Microsoft SQL Server 2005 Business Intelligence—Implementation and Maintenance* (Microsoft Press, ISBN 978-0-7356-9156-8). Based in Atlanta, Teo maintains a personal Web site, *www.prologika.com*, where he blogs about Microsoft business intelligence.

DEJAN SARKA focuses on development of database and BI applications. In addition to projects, he spends about half of his time on training and mentoring. He is a frequent speaker at some of the most important international conferences, such as PASS, TechEd, and SqlDevCon. He is also indispensable at regional Microsoft events—for example, at the NT Conference, the biggest Microsoft conference in Central and Eastern Europe. He is the founder of the Slovenian SQL Server and .NET Users Group. Dejan Sarka is the main author or coauthor of seven books about databases and SQL Server. Dejan Sarka also developed two courses for Solid Quality Mentors—Data Modeling Essentials and Data Mining with SQL Server 2008.

ALEJANDRO LEGUIZAMO is a SQL Server MVP and mentor for Solid Quality Mentors. He has been working with SQL Server since version 6.5 and with Microsoft Office Access since Office 97. His core area is data warehousing and BI, including ETL. He has wide experience in deployment and training in Business Intelligence Strategies, with a degree in Business Management strongly focused on Executive Information Systems. Alejandro often speaks at Microsoft events in Spain, Peru, Ecuador, and Venezuela.