Ray Rankins
Paul Bertucci
Chris Gallelli
Alex T. Silverstein

Microsoft® SQL Server 2005

UNLEASHED

SAMS

Ray Rankins,
Paul Bertucci,
Chris Gallelli,
Alex T. Silverstein,
et al.

Microsoft® SQL Server 2005

UNLEASHED

SAMS | 800 East 96th Street, Indianapolis, Indiana 46240 USA

Microsoft® SQL Server 2005 Unleashed

International Standard Book Number: 0-672-32824-0

Library of Congress Cataloging-in-Publication Data

Microsoft SQL server 2005 unleashed / Ray Rankins, et al.
p. cm.
ISBN 0-672-32824-0
1. SQL server. 2. Database management. I. Rankins, Ray.
QA76.9.D3M57365 2007
005.75'85—dc22

2007005947

Printed in the United States of America

First Printing: April 2007

10 09 08 07 4 3 2 1

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The authors and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the CD or programs accompanying it.

Bulk Sales

Sams Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

U.S. Corporate and Government Sales
1-800-382-3419
corpsales@pearsontechgroup.com

For sales outside of the U.S., please contact

International Sales
international@pearsoned.com

This Book Is Safari Enabled

The Safari® Enabled icon on the cover of your favorite technology book means the book is available through Safari Bookshelf. When you buy this book, you get free access to the online edition for 45 days.

Safari Bookshelf is an electronic reference library that lets you easily search thousands of technical books, find code samples, download chapters, and access technical information whenever and wherever you need it.

To gain 45-day Safari Enabled access to this book:

- Go to http://www.samspublishing.com/safarienabled
- Complete the brief registration form
- Enter the coupon code EYDB-XJWD-TEH7-QFE6-CG8Q

If you have difficulty registering on Safari Bookshelf or accessing the online edition, please e-mail customer-service@safaribooksonline.com.

Acquisitions Editor
Neil Rowe

Development Editor
Mark Renfrow

Managing Editor
Gina Kanouse

Project Editor
Andy Beaster

Copy Editor
Kitty Jarrett

Indexer
Lisa Stumpf

Proofreader
Paula Lowell

Technical Editor
Ross Mistry

Multimedia Developer
Dan Scherf

Book Designer
Gary Adair

Compositors
Bronkella Publishing
Nonie Ratcliff

Contents at a Glance

Table of Contents

About the Lead Authors

Ray Rankins is owner and president of Gotham Consulting Services, Inc. (www.gothamconsulting.com), near Saratoga Springs, New York. Ray has been working with Sybase and Microsoft SQL Server for more than 20 years and has experience in database administration, database design, project management, application development, consulting, courseware development, and training. He has worked in a variety of industries, including financial, manufacturing, health care, retail, insurance, communications, public utilities, and state and federal government. His expertise is in database performance and tuning, query analysis, advanced SQL programming and stored procedure development, database design, data architecture, and database application design and development. Ray's presentations on these topics at user group conferences have been very well received. Ray is coauthor of *Microsoft SQL Server 2000 Unleashed* (first and second editions), *Microsoft SQL Server 6.5 Unleashed* (all editions), *Sybase SQL Server 11 Unleashed*, and *Sybase SQL Server 11 DBA Survival Guide,* second edition, all published by Sams Publishing. He has also authored a number of articles, white papers, and database-related courses. As an instructor, Ray regularly teaches classes on SQL, advanced SQL programming and optimization, database design, database administration, and database performance and tuning. Ray's ability to bring his real-world experience into the classroom consistently rates very high marks from students in his classes for both his instructional skills and courseware. Ray can be reached at rrankins@gothamconsulting.com.

Paul Bertucci is the founder of Database Architechs (www.dbarchitechs.com), a database consulting firm with offices in the United States and Paris, France. He has more than 26 years of experience with database design, data architecture, data replication, performance and tuning, distributed data systems, data integration, high-availability assessments, and systems integration for numerous Fortune 500 companies, including Intel, 3COM, Coca-Cola, Apple, Toshiba, Lockheed, Wells Fargo, Safeway, Texaco, Charles Schwab, Cisco Systems, Sybase, Symantec, Veritas, and Honda, to name a few. He has authored numerous articles, standards, and high-profile courses, such as Sybase's "Performance and Tuning" and "Physical Database Design" courses. Other Sams Publishing books that he has authored include the highly popular *Microsoft SQL Server 2000 Unleashed*, *ADO.NET in 24 Hours*, and *Microsoft SQL Server High Availability.* He has deployed numerous systems with Microsoft SQL Server, Sybase, DB2, and Oracle database engines, and he has designed/architected several commercially available tools in the database, data modeling, performance and tuning, data integration, and multidimensional planning spaces. Paul's current working arrangement is as Symantec Corporations Chief Data Architect, and he also serves part time as chief technical advisor for a data integration server software company. Paul received his formal education in computer science and electrical engineering from UC Berkeley (Go Bears!). He lives in the great Pacific northwest (Oregon) with his wife, Vilay, and five children, Donny, Juliana, Paul Jr., Marissa, and Nina. Paul can be reached at pbertucci@DBArchitechs.com or Bertucci@Alum.CalBerkeley.Org.

Chris Gallelli is president of CGAL Consulting Services, Inc. His company focuses on consulting services in the areas of database administration, database tuning, and database programming using Visual Basic .NET. Chris has more than 10 years of experience with SQL Server and more than 20 years in the field of Information Technology. He has a Bachelor's degree in Electrical Engineering and a Masters in Business Administration from Union College. Chris was also one of the authors of *Microsoft SQL Server 2000 Unleashed* published by Sams Publishing. Chris currently lives near Albany, New York, with his lovely wife, Laura, and two daughters, Rachael and Kayla. You can contact Chris at CGallelli@gmail.com.

Alex T. Silverstein is managing principal and chief architect of the Unified Digital Group, LLC, a consulting and custom software development firm headquartered near Saratoga Springs, New York. He specializes in designing SQL Server and Microsoft .NET-powered solutions using the principles of Agile development and the Rational Unified Process. Alex has more than a decade of experience providing application development, database administration, and training services worldwide to a variety of industries. He was also a contributing author for *Microsoft SQL Server 2000 Unleashed* published by Sams Publishing. You can reach Alex anytime at alex@unifieddigital.com.

About the Contributing Authors

Tudor Trufinescu joined Microsoft in January 1999, and has since worked on a number of projects and technologies, including Microsoft Metadata services, HTTP-DAV, and SQL Server Reporting Services. He was one of the founding members of the SQL Server reporting services team. His team designed and built the server components of Reporting Services and the report controls included in Visual Studio 2005. Before joining Microsoft, Tudor helped build a number of software solutions, including a workflow and document management application and a real-time production monitoring system. Tudor received his degree in Electronic Engineering and Information Technology from the University of Bucharest, Romania, in 1992. Tudor and his family currently live in Redmond, Washington.

Dedication

I would like to dedicate this book to my beautiful wife of 20 years, Elizabeth, and my son, Jason, for their continued love, support, and understanding during the long hours and lost weekends spent working on this book.

—Ray Rankins

Dedicated to my wife, Vilay, for whom I say "koin hug chow" *("I love you" in Thai/Laotian).*

—Paul Bertucci

This book is dedicated to my wife, Laura, and my two daughters, Rachael and Kayla. Keeping an active house quiet while I was working on this book was no small feat but they made it happen. They showed a great deal of patience and understanding during the entire process. Thank you!!!

—Chris Gallelli

I dedicate this work to my girlfriend, Ellen, who patiently endured my absence during the scores of hours devoted to it.

—Alex T. Silverstein

Acknowledgments

I would first like to thank my coauthors for their tireless efforts in helping to turn out a quality publication and their willingness to take on more work when needed to help keep things on track. I would also like to thank Neil Rowe at Sams Publishing for providing us the opportunity to write this book and for his patience in regard to our deadlines.

Most of all, I wish to thank my family, Elizabeth and Jason, for their patience and understanding during the long days and late nights spent working on this book when I should have been spending quality family time with them.

—Ray Rankins

With any writing effort, there is always a huge sacrifice of time that must be made to properly research, demonstrate, and describe leading-edge subject matter. The brunt of the burden usually falls on those many people who are near and very dear to me. With this in mind, I desperately need to thank my family for allowing me to encroach on many months of what should have been my family's "quality time."

However, with sacrifice also comes reward in the form of technical excellence and solid business relationships. Many individuals were involved in this effort, both directly and indirectly, starting with Jeff Brzycki, Jack McElreath, Emily Breuner-Jaquette, Jay Jones, Mark Johnson, Scott Smith, and Walter Kuketz. And special thanks this time must go to my colleagues in France, Yves Moison and Thierry Gerardin. Their expertise in and knowledge of SQL, performance and tuning, and high availability are unmatched. *Merci beaucoup!*

—Paul Bertucci

Writing a book of this size and scope requires a tremendous amount of time and dedication. The time and dedication applies not only to the authors who are writing the book but also to their family members as well. My wife and daughters were very understanding while I was holed up working on this book and that understanding helped make the book happen. My love and thanks to them.

I would also like to thank many of my clients who embraced SQL Server 2005 and adopted the product shortly after it was released. In particular, I would like to thank Ray McQuade and his company Spruce Computer Systems. Spruce has had tremendous success with SQL Server 2005 and they gave me some of the "real-world" experience that was invaluable in creating this book.

—Chris Gallelli

I am most grateful to those whom I am gifted with the opportunity to serve: You are my teachers on this path. To my fellow authors, clients, friends, and mentors: Without you, I would not be the person I am today. Particularly, I'd like to express my appreciation to the men and women at Thomson Learning and at HANYS for making the consulting life a pleasure (and for using the latest technology!); to the staff of Sams Publishing for putting this book together; to my co-authors for trusting in me and providing such an awesome opportunity; to my family, both original and chosen, for their unconditional love; and, finally, to that unknown power that continually drives me on to a better life. Thanks to all who gave me a chance to grow throughout this process.

—Alex T. Silverstein

We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

As a Senior Acquisitions Editor for Sams Publishing, I welcome your comments. You can email or write me directly to let me know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that I cannot help you with technical problems related to the topic of this book. We do have a User Services group, however, where I will forward specific technical questions related to the book.

When you write, please be sure to include this book's title and author as well as your name, email address, and phone number. I will carefully review your comments and share them with the author and editors who worked on the book.

Email: feedback@samspublishing.com

Fax: 317-428-3310

Mail: Neil Rowe
Senior Acquisitions Editor
Sams Publishing
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at www.samspublishing.com/register for convenient access to any updates, downloads, or errata that might be available for this book.

This page intentionally left blank

Introduction

It has been just over six years since SQL Server 2000 was released and just over five years since we published the first edition of *SQL Server 2000 Unleashed*. In that time, SQL Server has established itself as a robust and reliable database platform whose performance and scalability meet the implementation needs of businesses and corporations from simple desktop applications on up to enterprise-wide systems. A number of significant changes and enhancements in SQL Server 2005 further solidify its position in the marketplace as a robust enterprise-wide database system that can compete on the same level as the other major enterprise-wide database products, shifting firmly to providing a database engine foundation that can be highly available 7 days a week, 365 days a year.

One of the biggest challenges we faced when we wrote *SQL Server 2000 Unleashed* six years ago was providing comprehensive, in-depth, all-inclusive coverage of all the features of SQL Server 2000 within a single book. Doing the same for SQL Server 2005 was even more of a challenge. SQL Server 2005 was in development at Microsoft for more than 5 years and represents a major upgrade to SQL Server 2000. Many features of SQL Server 2000 have been replaced completely in SQL Server 2005 (for example, SQL Enterprise Manager and SQL Query Analyzer have been replaced by SQL Server Management Studio), while some have been completely re-architected (for example, Analysis Services), and most others have undergone significant improvements and enhancements. In addition, the number of SQL Server features and components has increased, and many of these features and components (for example, SQL Server Integration Services, Reporting Services, and .NET Framework integration) provide enough material to warrant their own separate titles. After nearly all of the chapters for this book had been completed, we realized we had significantly more information than could reasonably fit into a single book and we had to make some hard decisions as to what to include in print.

We decided that the main focus for the book is for it to provide detailed coverage of the core database server product and the day-to-day administrative and management aspects and tools of SQL Server 2005. We also wanted to be sure to provide extensive coverage of the new features of the SQL Server 2005 database engine, while also providing sufficient coverage of the new components of SQL Server, such as SQL Server Integration Services, Reporting Services, Web Services, and integration with the .NET Framework. We wanted to be sure to provide enough of the necessary information, tips, and guidelines to get you started in making use of these features. However, at the same time, there was a wealth of useful information on various SQL Server 2005 topics and features that had already been written that we didn't want to just simply discard, so we decided that we would include a bonus CD with this book that would contain these additional "bonus" chapters. The chapters included on the bonus CD are described later in this Introduction. Also, as in the past, all of our example scripts, databases, and other material are provided on the bonus CD as well. These, by themselves, offer much value and provide practical guidance on exactly how to create and manage complex SQL Server 2005 solutions.

Our other main goal when writing this book was for it to be more than just a syntax reference. SQL Server Books Online is a fine resource as a syntax reference. This book attempts to pick up where Books Online leaves off, by providing, in addition to syntax where necessary, valuable insight, tips, guidelines, and useful examples derived from our many years of experience working with SQL Server. Although we do provide the core, and sometimes advanced, syntax elements for the SQL commands discussed, SQL Server Books Online provides a much more extensive syntax reference than would make sense to try to duplicate here. As a matter of fact, at times, we may even direct you to Books Online for more detail on some of the more esoteric syntax options available for certain commands.

We hope that we have succeeded in meeting the goals we set out for this book and that it becomes an essential reference and source of expert information for you as you work with SQL Server 2005.

Who This Book Is For

This *Unleashed* book is intended for intermediate- to advanced-level users: for SQL Server administrators who want to understand SQL Server more completely to be able to effectively manage and administer their SQL Server environments, and for developers who want a more thorough understanding of SQL Server to help them write better Transact-SQL (T-SQL) code and develop more robust SQL Server applications. If you are responsible for analysis, design, implementation, support, administration, or troubleshooting of SQL Server 2005, this book provides an excellent source of experiential information for you. You can think of this as a book of applied technology. The emphasis is on the more complex aspects of the product, including using the new tools and features, server administration, query analysis and optimization, data warehousing, management of very large databases, ensuring high availability, and performance tuning.

This book is for both developers and SQL Server administrators who are new to SQL Server 2005 as well as those who are already familiar with SQL Server 2000. At the beginning of each chapter is a brief summary of the major changes or new features or capabilities of SQL Server 2005 related to that topic. If you are already familiar with SQL Server 2000, you can use this information to focus on the information in the chapters that covers the new features and capabilities in more detail.

This book is intended to provide a behind-the-scenes look into SQL Server, showing you what goes on behind the various wizards and GUI-based tools so you can learn what the underlying SQL commands are. Although the GUI tools can make your average day-to-day operations much simpler, every database administrator should learn the underlying commands to the tools and wizards to fully unlock the power and capabilities of SQL Server. Besides, you never know when you may have to manage a server through a telnet session with only a command-line query tool available.

What This Book Covers

The book is divided into the following sections:

- **Part I, "Welcome to Microsoft SQL Server"**—This section introduces you to the Microsoft SQL Server 2005 environment, the various editions of SQL Server that are available, and the capabilities of each edition in the various Windows environments. In addition, it provides an overview of and introduction to the new features found in SQL Server 2005, which are covered in more detail throughout rest of the book.
- **Part II, "SQL Server Tools and Utilities"**—This section covers the tools and utility programs that SQL Server 2005 provides for you to administer and manage your SQL Server environments. You'll find information on the various management tools you use on a daily basis, such as SQL Server Management Studio and the new SQLCMD command-line query tool, along with information on SQL Server Profiler. If you are not familiar with these tools, you should read this part of the book early on because these tools are often used and referenced throughout many of the other chapters in the book.
- **Part III, "SQL Server Administration"**—This section discusses topics related to the administration of SQL Server at the server level. It begins with an overview of what is involved in administering a SQL Server environment and then goes on to cover the tasks related to setting up and managing the overall SQL Server environment, including installing and upgrading to SQL Server 2005 as well as installing SQL Server 2005 clients. This section also includes coverage of security and user administration, database backup and restore, replication, and using the new Database Mail facility. Chapters on SQL Server clustering and SQL Server high availability provide some expert advice in these areas. Database mirroring and task scheduling and notification using SQL Server Agent are also discussed in this section.

- **Part IV, "SQL Server Database Administration"**—This section dives into the administrative tasks associated with creating and managing a SQL Server 2005 database, including the creation and management of database objects, such as tables, indexes, views, stored procedures, functions, and triggers. It also provides coverage of the new Database Snapshot feature of SQL Server 2005 as well as an overview of database maintenance.
- **Part V, "SQL Server Performance and Optimization"**—This section provides information to help you get the best performance out of SQL Server. It begins with a discussion on indexes and performance, one of the key items to understand to help ensure good database performance. It then builds on that information with chapters on query optimization and analysis, locking, database design and performance, and monitoring and optimization of SQL Server performance.
- **Part VI, "SQL Server Application Development"**—This section includes a comprehensive overview of what's new in T-SQL in SQL Server 2005. In addition, chapters in this section provide an overview for developing SQL Server applications within the .NET Framework, working with XML in SQL Server 2005, and SQL Server 2005's built-in Web Services capabilities.
- **Part VII, "SQL Server Business Intelligence Features"**—This section includes a comprehensive overview of SQL Server 2005's built-in business intelligence features: Analysis Services, Integration Services, and Reporting Services.
- **Bonus Chapters on the CD**—In order to be able to provide comprehensive coverage of the new features of SQL Server 2005 and still fit everything in a single book that doesn't require a wheelbarrow to transport, some information had to be omitted from the printed product. However, we have included this information as bonus chapters on the enclosed CD. Most of these bonus chapters cover additional SQL Server components that are not part of the core database engine such as Notification Services, Service Broker, and Full-Text Search. There are also chapters for which there just wasn't room enough to include in the book itself. These chapters provide expert advice and information on remote and linked server management, SQL Server configuration, tuning and optimization, administering very large SQL Server databases, SQL Server Disaster Recovery Planning, and T-SQL programming guidelines, tips, and tricks. In addition, please visit the web page for this book on www.samspublishing.com periodically for any updated or additional bonus material as it becomes available.
- **Book Materials on the CD**—Also included on the CD are many of the code samples, scripts, databases, and other materials that supplement various chapters. This has always been one of the most valuable reasons to buy the *Unleashed* series books. It is our goal to not just discuss a SQL technique or solution, but to also provide working samples and examples that actually do it. Learning by seeing is essential for understanding.

Conventions Used in This Book

Names of commands and stored procedures are presented in a special `monospaced` computer typeface. We have tried to be consistent in our use of uppercase and lowercase for keywords and object names. However, because the default installation of SQL Server doesn't make a distinction between upper- and lowercase for SQL keywords or object names and data, you might find some of the examples presented in either upper- or lowercase.

Code and output examples are presented separately from regular paragraphs and are also in a `monospaced` computer typeface. The following is an example:

```
select object_id, name, type_desc
from sys.objects
where type = 'SQ'

object_id   name                           type_desc
----------- ------------------------------ ------------
1977058079  QueryNotificationErrorsQueue   SERVICE_QUEUE
2009058193  EventNotificationErrorsQueue   SERVICE_QUEUE
2041058307  ServiceBrokerQueue             SERVICE_QUEUE
```

When syntax is provided for a command, we have followed these conventions:

Syntax Element	Definition
`command`	These are command names, options, and other keywords.
`placeholder`	Monospaced italic indicates values you provide.
`{}`	You must choose at least one of the enclosed options.
`[]`	The enclosed value/keyword is optional.
`()`	Parentheses are part of the command.
`¦`	You can select only one of the options listed.
`,`	You can select any of the options listed.
`[...]`	The previous option can be repeated.

Consider the following syntax example:

```
grant {all ¦ permission_list} on object [(column_list)]
       to {public ¦ user_or_group_name [, [...]]}
```

In this case, *`object`* is required, but *`column_list`* is optional. Note also that items shown in plain computer type, such as `grant`, `public`, and `all`, should be entered literally, as shown. Placeholders are presented in *`italic`*, such as *`permission_list`* and *`user_or_group_name`*. A *placeholder* is a generic term for which you must supply a specific value or values. The ellipsis (`[...]`) in the square brackets following *`user_or_group_name`* indicates

that multiple user or group names can be specified, separated by commas. You can specify either the keyword `public` or one or more user or group names, but not both.

Some of the examples presented in this book make use of the `AdventureWorks` database, which is included with SQL Server 2005 (our old friends the `pubs` and `Northwinds` databases are no longer provided with SQL Server). However, for many of the examples presented in Part V, larger tables than what are available in the `Adventureworks` database were needed to demonstrate many of the concepts with more meaningful examples. For many of the chapters in this section, the examples come from the `bigpubs2005` database. This database has the same structure as the old `pubs` database, but it contains significantly more data. A copy of the database, along with an Entity-Relationship (ER) diagram and table descriptions, is also on the CD.

To install the `bigpubs2005` database on your system so you can try out the various examples, do the following:

1. Copy the `bigpubs2005.mdf` file into the SQL Server data folder where you want it to reside.
2. After the file has been copied to the destination folder, ensure that the `Read-Only` property of the `bigpubs2005.mdf` file is not enabled (this can happen when the file is copied from the CD). Right-click the file in Windows Explorer and select Properties to bring up the Properties dialog. Click the Read-Only check box to remove the check mark. Click OK to save the changes to the file attributes.
3. Attach the `bigpubs2005` database by using a command similar to the following:

   ```
   sp_attach_single_file_db 'bigpubs2005',
       N'D:\MSSQL\DATA\MSSQL.1\MSSQL\Data\bigpubs2005.mdf'
   ```

   Note that you might need to edit the path to match the location where you copied the `bigpubs2005.mdf` file.

Alternatively, you can attach the database by using SQL Server Management Studio. You right-click the `Databases` node in the Object Explorer and select Attach. When the Attach Databases dialog appears, click the Add button, locate the `bigpubs2005.mdf` file, and click OK. In the bottom window pane, click the transaction log file entry (it should say `Not Found` in the message column) and click the Remove button. Next, click the OK button to attach the database. A new transaction log file is automatically created in the same folder as the `bigpubs2005.mdf` file. For more information on attaching database files, see Chapters 11, “Database Backup and Restore,” and 18, “Creating and Managing Databases.”

NOTE

In addition to the `bigpubs2005` database, the mdf file for the database that is used for examples in Chapter 39, "SQL Server Analysis Services," (CompSales) is also provided. To install the `CompSales` database, do the following:

1. Copy the `CompSales.mdf` file into the SQL Server data folder where you want it to reside.
2. Ensure that the `Read-Only` property of the `CompSales.mdf` file is not enabled.
3. Attach the `CompSales` database by using a command similar to the following (edit the path to match the location of the `CompSales.mdf` file on your system):

```
sp_attach_single_file_db 'CompSales',
     N'D:\MSSQL\DATA\MSSQL.1\MSSQL\Data\CompSales.mdf'
```

Good Luck!

If you have purchased this book, you are on your way to getting the most from SQL Server 2005. You have already chosen a fine platform for building database applications, one that can provide outstanding performance and rock-solid reliability and availability at a reasonable cost. With this book, you now have the information you need to make the best of it.

Many of us who worked on this book have been using SQL Server for more than a decade. Writing about this new version challenged us to reassess many of our preconceived notions about SQL Server and the way it works. It was an interesting and enjoyable process, and we learned a lot. We hope you get as much enjoyment and knowledge from reading this book as we have from writing it.

This page intentionally left blank

PART I

Welcome to Microsoft SQL Server

IN THIS PART

This page intentionally left blank

CHAPTER 1

SQL Server 2005 Overview

IN THIS CHAPTER

- SQL Server Components and Features
- SQL Server 2005 Editions
- SQL Server Licensing Models

Exactly what is SQL Server 2005? When you first install the product, what are all the pieces you get, what do they do, and which of them do you need?

At its core, SQL Server 2005 is an enterprise-class database management system (DBMS) that is capable of running anything from a personal database only a few megabytes in size on a handheld Windows Mobile device up to a multi-server database system managing terabytes of information. However, SQL Server 2005 is much more than just a database engine.

The SQL Server product is made up of a number of different components. This chapter describes each of the pieces that make up the SQL Server product and what role each plays. Each of these topics is dealt with in more detail later in the book. In addition, this chapter looks at the environments that support SQL Server 2005 and the features available in each of the various SQL Server editions.

SQL Server Components and Features

The main component of SQL Server 2005 is the Database Engine. Before you can use the other components and features of SQL Server 2005, which are discussed in this section, you need to have an instance of the Database Engine installed.

The SQL Server Database Engine

The Database Engine is the core application service in the SQL Server package for storing, processing, and securing data with SQL Server 2005. The SQL Server 2005 Database

Engine is a Windows service that can be used to store and process data in either a relational format or as XML documents. The following are the main responsibilities of the Database Engine:

- Provide reliable storage for data
- Provide a means to rapidly retrieve this data
- Provide consistent access to the data
- Control access to the data through security
- Enforce data integrity rules to ensure that the data actually means something

Each of these responsibilities is examined in greater detail in later chapters in this book. For now, this chapter provides just a brief overview on each of these points to show how Microsoft SQL Server fulfills these core responsibilities.

Reliable Storage

Reliable storage starts at the hardware level. This isn't the responsibility of the Database Engine, but it's a necessary part of a well-built database. Although you can put an entire SQL database on a single IDE or SATA drive (or even burn a read-only copy on a CD), it is preferable to maintain the data on RAID arrays. The most common RAID arrays allow hardware failures at the disk level without loss of data.

NOTE

For more information on the reliability characteristics and performance implications of the various RAID configurations and guidelines for implementing RAID configurations with SQL Server, see Chapter 33, "Database Design and Performance."

Using whatever hardware you have decided to make available, the Database Engine manages all the data structures necessary to ensure reliable storage of your data. Rows of data are stored in *pages*, and each page is 8KB in size. Eight pages make up an *extent,* and the Database Engine keeps track of which extents are allocated to which tables and indexes.

NOTE

A *page* is an 8KB chunk of a data file, the smallest unit of storage available in the database. An *extent* is a collection of eight 8KB pages.

Another key feature the Database Engine provides to ensure reliable storage is the transaction log. The transaction log makes a record of every change that is made to the database. For more information on the transaction log and how it's managed, see Chapter 26, "Transaction Management and the Transaction Log."

NOTE

It is not strictly true that the transaction log records *all* changes to the database; some exceptions exist. Operations on binary large objects—data of type `image` and `text`—can be excepted from logging, and bulk copy loads into tables can be minimally logged to get the fastest possible performance.

Rapid Data Access

SQL Server allows the creation of indexes, enabling fast access to data. See Chapter 29, "Indexes and Performance," for an in-depth discussion of indexes.

Another way to provide rapid access to data is to keep frequently accessed data in memory. Excess memory for a SQL Server instance is used as a data cache. When pages are requested from the database, the SQL Server Database Engine checks to see if the requested pages are already in the cache. If they are not, it reads them off the disk and stores them in the data cache. If there is no space available in the data cache, the least recently accessed pages (that is, those that haven't been accessed in a while, since they were read into memory) are flushed out of the data cache to make room for the newly requested pages. If the pages being flushed contain changes that haven't been written out yet, they are written to disk before being flushed from memory. Otherwise, they are simply discarded.

NOTE

With sufficient memory, an entire database can fit completely into memory, providing the best possible I/O performance for the database.

Consistent Data Access

Getting to your data quickly doesn't mean much if the information you receive is inaccurate. SQL Server follows a set of rules to ensure that the data you receive from queries is consistent.

The general idea with consistent data access is to allow only one client at a time to change the data and to prevent others from reading data from the database while it is undergoing changes. Data and transactional consistency are maintained in SQL Server by using transactional locking.

Transactional consistency has several levels of conformance, each of which provides a trade-off between accuracy of the data and concurrency. These levels of concurrency are examined in more detail in Chapter 32, "Locking and Performance."

Access Control

SQL Server controls access by providing security at multiple levels. Security is enforced at the server level, at the database level, and at the object level. In SQL Server 2005, security can now also be enforced at the schema level.

Sever-level access is enforced either by using a local user name and password or through integrated network security, which uses the client's network login credentials to establish identity.

SQL Server security is examined in greater detail in Chapter 10, "Security and User Administration."

Data Integrity

Some databases have to serve the needs of more than a single application. A corporate database that contains valuable information might have a dozen different departments wanting to access portions of the database for different needs.

In this kind of environment, it is impractical to expect the developers of each application to agree on an identical set of standards for maintaining data integrity. For example, one department might allow phone numbers to have extensions, whereas another department may not need that capability. One department might find it critical to maintain a relationship between a customer record and a salesperson record, whereas another might care only about the customer information.

The best way to keep everybody sane in this environment—and to ensure that the data stays consistent and usable by everyone—is to enforce a set of data integrity rules within the database itself. This is accomplished through data integrity constraints and other data integrity mechanisms, such as rules, defaults, and triggers. See Chapter 21, "Implementing Data Integrity," for details.

SQL Server 2005 Administration and Management Tools

SQL Server 2005 provides a suite of tools for managing and administering the SQL Server Database Engine and other components. This section provides an overview of the primary tools for day-to-day administration, management, and monitoring of your SQL Server environments.

SQL Server Management Studio (SSMS)

SSMS is the central console from which most database management tasks can be coordinated. SSMS provides a single interface from which all servers in a company can be managed. SSMS is examined in more detail in Chapter 3, "SQL Server Management Studio."

Figure 1.1 shows SSMS being used for some everyday administration tasks.

Figure 1.1 shows a list of registered servers in the upper-left pane. Below that is the Object Explorer, which lets you browse the contents of the databases within a SQL Server instance. The `bigpubs2005` database has been expanded, and the right pane shows the columns for the `authors` table.

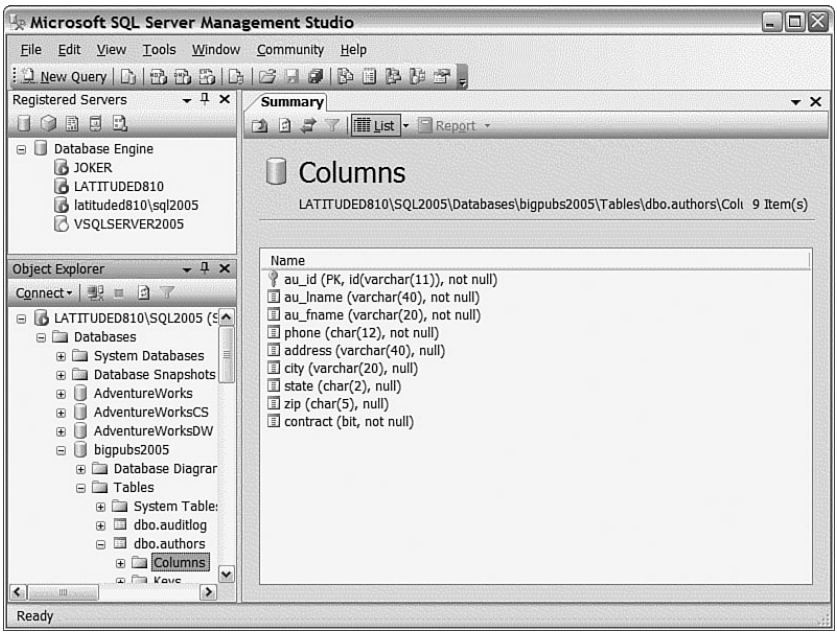

FIGURE 1.1 SSMS, showing a list of columns for the `authors` table in the `bigpubs2005` database.

The following are some of the tasks you can perform with SSMS. Most of these are discussed in detail later in the book:

- Completely manage many servers in a convenient interface
- Set server options and configuration values, such as the amount of memory and number of processors to use, the default language, and the default location of the data and log files
- Manage logins, database users, and database roles
- Create, edit, and schedule automated jobs through the SQL Server Agent
- Back up and restore databases and define maintenance plans
- Create new databases
- Browse table contents
- Manage database objects, such as tables, indexes, and stored procedures
- Generate DDL scripts for databases and database objects
- Configure and manage replication
- Create, edit, and analyze Transact-SQL (T-SQL) scripts

- Manage and organize scripts into projects and save versions in source control systems such as Visual SourceSafe

NOTE

SSMS interacts with SQL Server by using plain old T-SQL commands. For example, when you create a new database through the SSMS interface, behind the scenes, SSMS generates a `CREATE DATABASE` SQL command. Whatever you can do through the SSMS GUI, you can do with T-SQL statements.

If you're curious about how SSMS is accomplishing something, you can run SQL Profiler to capture the commands that SSMS is sending to the server. You can use this technique to discover some interesting internals information. In addition, almost every dialog in SSMS provides the ability to generate a T-SQL script for any actions it performs.

The SQL Server Surface Area Configuration Tool

SQL Server 2005 ships with a number of features and components, some if which you might not need. Stopping or disabling unused components helps to improve security by providing fewer avenues for potential attacks on a system as well as reducing requirements for CPU and memory resources.

You can use the SQL Server Surface Area Configuration tool to verify which SQL Server features and services are enabled and running and to verify which types of connections SQL Server 2005 accepts. You can use this tool to verify or change the state of features, services, and connections.

SQL Server Configuration Manager

SQL Server Configuration Manager is a new tool provided with SQL Server 2005 for managing the services associated with SQL Server and for configuring the network protocols used by SQL Server. SQL Server Configuration Manager provides the same functionality as the multiple tools provided with SQL Server 2000: Server Network Utility, Client Network Utility, and Service Manager. Primarily, SQL Server Configuration Manager is used to start, pause, resume, and stop SQL Server services and to view or change service properties.

SQL Server Agent

SQL Server Agent is a scheduling tool integrated into SSMS that allows convenient definition and execution of scheduled scripts and maintenance jobs. SQL Server Agent also handles automated alerts—for example, if the database runs out of space.

SQL Server Agent is a Windows service that runs on the same machine as the SQL Server Database Engine. The SQL Server Agent service can be started and stopped through either SSMS, the SQL Server Configuration Manager, or the ordinary Windows Services Manager.

In enterprise situations in which many SQL Server machines need to be managed together, the SQL Server Agent can be configured to distribute common jobs to multiple

servers through the use of Multiserver Administration. This is most helpful in a wide architecture scenario, in which many SQL Server instances are performing the same tasks with the databases. Jobs are managed from a single SQL Server machine, which is responsible for maintaining the jobs and distributing the job scripts to each target server. The results of each job are maintained on the target servers but can be observed through a single interface.

If you had 20 servers that all needed to run the same job, you could check the completion status of that job in moments instead of logging in to each machine and checking the status 20 times.

The SQL Server Agent also handles event forwarding. Any system events that are recorded in the Windows system event log can be forwarded to a single machine. This gives a busy administrator a single place to look for errors.

More information about how to accomplish these tasks, as well as other information on the SQL Server Agent, is available in Chapter 13, "SQL Server Scheduling and Notification."

SQL Server Profiler

The SQL Server Profiler is a GUI interface to the SQL Trace feature of SQL Server that captures the queries and results flowing to and from the database engine. It is analogous to a network sniffer, although it does not operate on quite that low a level. The Profiler has the ability to capture and save a complete record of all the T-SQL statements passed to the server and the occurrence of SQL Server events such as deadlocks, logins, and errors. You can use a series of filters to pare down the results when you want to drill down to a single connection or even a single query.

You can use the SQL Profiler to perform these helpful tasks:

- You can capture the exact SQL statements sent to the server from an application for which source code is not available (for example, third-party applications).
- You can capture all the queries sent to SQL Server for later playback on a test server. This is extremely useful for performance testing with live query traffic.
- If your server is encountering recurring access violations (AVs), you can use Profiler to reconstruct what happened leading up to an AV.
- The Profiler shows basic performance data about each query. When your users start hammering your server with queries that cause hundreds of table scans, the Profiler can easily identify the culprits.
- For complex stored procedures, the Profiler can identify which portion of the procedure is causing the performance problem.
- You can audit server activity in real-time.

More information on SQL Server Profiler is available in Chapter 5, "SQL Server Profiler."

Replication

Replication is a server-based tool that you can use to synchronize data between two or more databases. Replication can send data from one SQL Server instance to another, or it can include Oracle, Access, or any other database that is accessible via ODBC or OLE DB.

SQL Server supports three kinds of replication:

- Snapshot replication
- Transactional replication
- Merge replication

The availability and functionality of replication might be restricted, depending on the version of SQL Server 2005 you are running.

NOTE

Replication copies the data from your tables and indexed views and even replicates changes to multiple tables caused by a stored procedure, but it does not normally re-create indexes or triggers at the target. It is common to have different indexes on replication targets rather than on the source to support different requirements.

Snapshot Replication

With snapshot replication, the server takes a picture, or snapshot, of the data in a table at a single point in time. Usually, if this operation is scheduled, the target data is simply replaced at each update. This form of replication is appropriate for small data sets, infrequent update periods (or for a one-time replication operation), or management simplicity.

Transactional Replication

Initially set up with a snapshot, the server maintains downstream replication targets by reading the transaction log at the source and applying each change at the targets. For every insert, update, and delete operation, the server sends a copy of the operation to every downstream database. This is appropriate if low-latency replicas are needed. Transactional replication can typically keep databases in sync within about five seconds of latency, depending on the underlying network infrastructure. Keep in mind that transactional replication does not guarantee identical databases at any given point in time. Rather, it guarantees that each change at the source will eventually be propagated to the targets. If you need to guarantee that two databases are transactionally identical, you should look into Distributed Transactions or Database Mirroring.

Transactional replication might be used for a website that supports a huge number of concurrent browsers but only a few updaters, such as a large and popular messaging board. All updates would be done against the replication source database and would be replicated in near-real-time to all the downstream targets. Each downstream target could support several web servers, and each incoming web request would be balanced among

the web farm. If the system needed to be scaled to support more read requests, you could simply add more web servers and databases and add the database to the replication scheme.

Merge Replication

With snapshot and transactional replication, a single source of data exists from which all the replication targets are replenished. In some situations, it might be necessary or desirable to allow the replication targets to accept changes to the replicated tables and merge these changes together at some later date.

Merge replication allows data to be modified by the subscribers and synchronized at a later time. This could be as soon as a few seconds, or it could be a day later.

Merge replication would be helpful for a sales database that is replicated from a central SQL Server database out to several dozen sales laptops. As the sales personnel make sales calls, they can add new data to the customer database or change errors in the existing data. When the salespeople return to the office, they can synchronize their laptops with the central database. Their changes are submitted, and the laptops get refreshed with whatever new data was entered since the last synchronization.

Immediate Updating

Immediate updating allows a replication target to immediately modify data at the source. This is accomplished by using a trigger to run a distributed transaction. Immediate updating is performance intensive, but it allows for updates to be initiated from anywhere in the replication architecture.

More details on replication are available in Chapter 15, "Replication."

Database Mirroring

Database mirroring is a new feature available in SQL Server 2005. Database mirroring is primarily a software solution for increasing database availability. Essentially, database mirroring maintains two copies of a single database that reside on different instances of SQL Server, typically on server instances that reside on computers in different locations. In a typical database mirroring scenario, one server instance serves as the primary database to which the client applications connect, and the other server instance acts as a hot or warm standby server.

Database mirroring involves redoing every modification operation that occurs on the primary database onto the mirror database as quickly as possible. This is accomplished by sending every active transaction log record generated on the primary server to the mirror server. The log records are applied to the mirror database, in sequence, as quickly as possible. Unlike replication, which works at the logical level, database mirroring works at the level of the physical log record. The mirror database is an exact copy of the primary database.

For more information on setting up and using database mirroring, see Chapter 16, "Database Mirroring."

Full-Text Search

SQL Server 2005 provides the ability to issue full-text queries against plain character-based data in your SQL Server tables. This is useful for searching large text fields, such as movie reviews, book descriptions, or case notes. Full-text queries can include words and phrases, or multiple forms of a word or phrase.

Full-Text Search capabilities in Microsoft SQL Server 2005 are provided by the Microsoft Full-Text Engine for SQL Server (MSFTESQL). The MSFTESQL service works together with the SQL Server Database Engine. You specify tables or entire databases that you want to index. The full-text indexes are built and maintained outside the SQL Server database files in special full-text indexes stored in the Windows file system. You can specify how often the full-text indexes are updated to balance performance issues with timeliness of the data.

NOTE

The MSFTESQL service is a separate service from the SQL Server Database Engine service. You can enable or disable the `MSFTESQL` service by using the SQL Server 2005 Surface Area Configuration tool.

The SQL Server Database Engine supports basic text searches against specific columns. For example, to find all the rows where a text column contained the word *guru*, you might write the following SQL statement:

```
select *
   from resume
   where description like '%guru%'
```

This finds all the rows in the `resume` table where the description contains the word *guru*. This method has a couple problems, however. First, the search is slow. Because the Database Engine can't index text columns, a full table scan has to be done to satisfy the query. Even if the data were stored in a `varchar` column instead of a `text` column, an index may not help because you're looking for *guru* anywhere in the column, not just at the beginning. (More information on avoiding situations like this are discussed in Chapter 29, "Indexes and Performance.")

What if you wanted to search for the word *guru* anywhere in the table, not just in the description column? What if you were looking for a particular set of skills, such as "SQL" and "ability to work independently"? Full-text indexing addresses these problems. To perform the same search as before with full-text indexing, you might use a query like this:

```
select *
   from resume
   where contains(description, 'guru')
```

To perform a search that looks for a set of skills, you might use a query like this:

```
select *
   from resume
   where contains(*, 'SQL and "ability to work independently"')
```

For more information on setting up and searching Full-Text Search indexes, see Chapter 49, "SQL Server Full-Text Search" (on the CD-ROM).

SQL Server Integration Services (SSIS)

SSIS is a platform for building high-performance data integration solutions and workflow solutions. You can build extract, transform, and load (ETL) packages to update data warehouses, interact with external processes, clean and mine data, process analytical objects, and perform administrative tasks. In SQL Server 2000, these tasks were performed by Data Transformation Services (DTS). In SQL Server 2005, Microsoft has completely redeployed and rebuilt DTS into SSIS and integrated it into the Business Intelligence (BI) Development Studio/Visual Studio development environments and SSMS.

The following are some of the features of SSIS:

- Graphical tools and wizards for building, debugging, and deploying SSIS packages
- Workflow functions, such as File Transfer Protocol (FTP), SQL statement execution, and more
- SSIS application programming interfaces (APIs)
- Complex data transformation for data cleansing, aggregation, merging, and copying
- An email messaging interface
- A service-based implementation
- Support for both native and managed code (C++ or any common language runtime [CLR]–compliant language, such as C# or J#)
- An SSIS object model

SSIS is a tool that helps address the needs of getting data—which is often stored in many different formats, contexts, file systems, and locations—from one place to another. In addition, the data often requires significant transformation and conversion processing as it is being moved around. Common uses of SSIS might include the following:

- Exporting data out of SQL Server tables to other applications and environments (for example, ODBC or OLE DB data sources, flat files)
- Importing data into SQL Server tables from other applications and environments (for example, ODBC or OLE DB data sources, flat files)
- Initializing data in some data replication situations, such as initial snapshots

- Aggregating data (that is, data transformation) for distribution to/from data marts or data warehouses
- Changing the data's context or format before importing or exporting it (that is, data conversion)

For more information on creating and using SSIS packages, see Chapter 40, "SQL Server Integration Services."

SQL Server Analysis Services (SSAS)

SSAS provides online analytical processing (OLAP) and data mining functionality for BI solutions. SSAS provides a rich set of data mining algorithms to enable business users to mine data, looking for specific patterns and trends. These data mining algorithms can be used to analyze data through a Unified Dimensional Model (UDM) or directly from a physical data store.

SSAS uses both server and client components to supply OLAP and data mining functionality for BI applications. SSAS consists of the analysis server, processing services, integration services, and a number of data providers. SSAS has both server-based and client-/local-based analysis services capabilities. This essentially provides a complete platform for SSAS. The basic components within SSAS are all focused on building and managing data cubes.

SSAS allows you to build dimensions and cubes from heterogeneous data sources. It can access relational OLTP databases, multidimensional data databases, text data, and any other source that has an OLE DB provider available. You don't have to move all your data into a SQL Server 2005 database first; you just connect to its source. In addition, SSAS allows a designer to implement OLAP cubes, using a variety of physical storage techniques that are directly tied to data aggregation requirements and other performance considerations.

You can easily access any OLAP cube built with SSAS via the Pivot Table Service, you can write custom client applications by using Multidimensional Expressions (MDX) with OLE DB for OLAP or ActiveX Data Objects Multidimensional (ADO MD), and you can use a number of third-party OLAP-compliant tools. MDX enables you to formulate complex multidimensional queries.

SSAS is commonly used to perform the following tasks:

- Perform trend analysis to predict the future. For example, based on how many widgets you sold last year, how many will you sell next year?
- Combine otherwise disconnected variables to gain insight into past performance. For example, was there any connection between widget sales and rainfall patterns? Searching for unusual connections between your data points is a typical data mining exercise.

- Perform offline summaries of commonly used data points for instant access via a web interface or a custom interface. For example, a relational table might contain one row for every click on a website. OLAP can be used to summarize these clicks by hour, day, week, and month and then to further categorize these by business line.

SSAS is a complex topic. For more information on MDX, data cubes, and how to use data warehousing analysis services, see Chapter 39, "SQL Server 2005 Analysis Services."

SQL Server 2005 Reporting Services

SQL Server 2005 Reporting Services is a server-based reporting platform that delivers enterprise, web-enabled reporting functionality so you can create reports that draw content from a variety of data sources, publish reports in various formats, and centrally manage security and subscriptions.

Reporting Services includes the following core components:

- A complete set of tools you can use to create, manage, and view reports
- A report server component that hosts and processes reports in a variety of formats, including HTML, PDF, TIFF, Excel, CSV, and more
- An API that allows developers to integrate or extend data and report processing into custom applications or to create custom tools to build and manage reports

There are two design tools for building reports: BI Development Studio, a powerful development tool integrated with Visual Studio .NET 2005, and Report Builder, which is a simpler point-and-click tool that you use to design ad hoc reports. Both report design tools provide a WYSIWYG experience.

Reports are described by Report Definition Language (RDL). RDL contains the description of the report layout, formatting information, and instructions on how to fetch the data. It can optionally contain custom code written in VB .NET that is executed as part of the report.

After a report is defined, it can be deployed on the report server, where it can be managed, secured, and delivered to a variety of formats, including HTML, Excel, PDF, TIFF, and XML. Various delivery, caching, and execution options are also available, as are scheduling and historical archiving.

For more information on designing and deploying reports using Reporting Services, see Chapter 41, "SQL Server 2005 Reporting Services."

SQL Server Notification Services

SQL Server Notification Services is an environment for developing and deploying applications that generate and send notifications. You can use Notification Services to generate and send timely, personalized messages to thousands or millions of subscribers, and you

can deliver the messages to a variety of devices, including mobile phones, personal digital assistants (PDAs), Microsoft Windows Messenger, or email accounts.

Notification Services consists of the following:

- A Notification Services programming framework that enables you to quickly create and deploy notification applications by using XML or Notification Services Management Objects (NMO)
- A reliable, high-performance, scalable Notification Services engine that runs notification applications

In order to receive notifications, subscribers create subscriptions to notification applications. A *subscription* is an expressed interest in a specific type of event, such as when a stock price reaches a specified price or a when a document has been updated. Notifications are generated and sent to the subscriber when a triggering event occurs or notifications can be generated and sent on a predetermined schedule specified by the subscriber.

For more information on building and deploying notification applications using Notification Services, see Chapter 47, "SQL Server Notification Services" (on the CD-ROM).

SQL Server Service Broker

SQL Server Service Broker is a new feature in SQL Server 2005. Service Broker provides a native SQL Server infrastructure that supports asynchronous, distributed messaging between database-driven services. Service Broker handles all the hard work of managing coordination among the constructs required for distributed messaging, including transactional delivery and storage, message typing and validation, multithreaded activation and control, event notification, routing, and security.

Service Broker is designed around the basic functions of sending and receiving messages. An application sends messages to a *service*, which is a name for a set of related tasks. An application receives messages from a *queue*, which is a view of an internal table. Service Broker guarantees that an application receives each message exactly once, in the order in which the messages were sent.

Service Broker can be useful for any application that needs to perform processing asynchronously or that needs to distribute processing across a number of computers. An example would be a bicycle manufacturer and seller who must provide new and updated parts data to a company that implements a catalog management system. The manufacturer must keep the catalog information up-to-date with its product model data, or it could lose market share or end up receiving orders from distributors based on out-of-date catalog information. When the parts data is updated in the manufacturer's database, a trigger could be invoked to send a message to Service Broker with information about the updated data. Service Broker would then asynchronously deliver the message to the catalog service. The catalog service program would then perform the work in a separate transaction. By performing this work in a separate transaction, the original transaction in the manufacturer's database can commit immediately. The application avoids system

slowdowns that result from keeping the original transaction open while performing the update to the catalog database.

For more information on using Service Broker, see Chapter 48, "SQL Server Service Broker" (on the CD-ROM).

SQL Server 2005 Editions

You can choose from several editions of SQL Server 2005. The edition you choose depends on your database and data processing needs, as well as the Windows platform on which you want to install it.

For actual deployment of SQL Server in a production environment, you can choose from any edition of SQL Server 2005 except Developer Edition and Evaluation Edition. Which edition you choose to deploy depends on your system requirements and need for SQL Server components.

This chapter examines the different editions of SQL Server and discusses their features and capabilities. Using this information, you can better choose which edition provides the appropriate solution for you.

SQL Server 2005 Standard Edition

The Standard Edition of SQL Server 2005 is the version intended for the masses—those running small- to medium-sized systems who don't require the performance, scalability, and availability provided by Enterprise Edition. Standard Edition runs on any of the Windows 2000 or Windows 2003 Server platforms, and its scalability is limited to up to four processors. There is no built-in memory limitation in SQL Server 2005 Standard Edition as there was in SQL Server 2000; it can utilize as much memory as provided by the operating system.

SQL Server 2005 Standard Edition includes the following features:

- CLR procedures, functions, and data types
- SQL Server Analysis Services
- Service Broker
- Reporting Services
- Notification Services
- SQL Server Integration Services
- Full-Text Search
- Built-in XML support
- SQL Server Profiler and performance analysis tools
- SQL Server Management Studio

- Replication
- Two-node failover clustering
- Database mirroring (safety full mode only)
- Log shipping

The Standard Edition can be installed on any of the Windows 2000 and Windows 2003 Server platforms, as well as Windows XP.

The Standard Edition should meet the needs of most departmental and small- to midsized applications. However, if you need more scalability, availability, advanced performance features, or comprehensive analysis features, you should implement the Enterprise Edition of SQL Server 2005.

SQL Server 2005 Enterprise Edition

The Enterprise Edition of SQL Server 2005 is the most comprehensive and complete edition available. It provides the most scalability and availability of all editions and is intended for systems that require high performance and availability, such as large-volume websites, data warehouses, and high-throughput online transaction processing (OLTP) systems.

SQL Server 2005 Enterprise Edition supports as much memory and as many CPUs as supported by the operating system it is installed on. It can be installed on any of the Windows 2000 and Windows 2003 server platforms.

In addition, SQL Server 2005 Enterprise Edition provides performance enhancements, such as parallel queries, indexed views, and enhanced read-ahead scanning.

Which version is right for you? The next section explores the feature sets of Enterprise and Standard Editions so you can decide which one provides the features you need.

Differences Between the Enterprise and Standard Editions of SQL Server

For deploying SQL Server 2005 in a server environment, either the Standard Edition or the Enterprise Edition of SQL Server is a logical choice. To help decide between the two editions, Table 1.1 compares the major features that each edition supports.

TABLE 1.1 SQL Server 2005 Feature Comparison: Enterprise and Standard Editions

Feature	Enterprise Edition	Standard Edition
Max number of CPUs	Unlimited	4
64 bit support	Yes	Yes
CLR runtime integration	Yes	Yes
Full-Text Search	Yes	Yes
SQL Server Integration Services	Yes	Yes
Integration Services with Basic Transforms	Yes	Yes

TABLE 1.1 Continued

Feature	Enterprise Edition	Standard Edition
Integration Services with Advanced data mining and cleansing transforms	Yes	No
Service Broker	Yes	Yes
Notification Services	Yes	Yes
Reporting Services	Yes	Yes
Replication	Yes	Yes
Log shipping	Yes	Yes
Database Mirroring	Yes	Yes (Single REDO thread with Safety FULL only)
Database snapshot	Yes	No
Indexed views	Yes	Yes (Can be created but automatic matching by Query Optimizer not supported)
Updatable distributed partitioned views	Yes	No
Table and index partitioning	Yes	No
Online index operations	Yes	No
Parallel index operations	Yes	No
Parallel DBCC	Yes	No
Online page and file restoration	Yes	No
Fast Recovery	Yes	No
Failover clustering	Yes	Yes (2-node only)
Multiple-instance support	Yes (50 instances max.)	Yes (16 instances max.)

Other SQL Server 2005 Editions

The Standard and Enterprise Editions of SQL Server 2005 are intended for server-based deployment of applications. In addition, the following editions are available for other specialized uses:

- Workgroup Edition
- Developer Edition
- Express Edition
- Mobile Edition

Workgroup Edition

SQL Server 2005 Workgroup Edition is intended for small organizations that need a database with no limits on database size or number of users but may not need the full capabilities of the Standard Edition. SQL Server 2005 Workgroup Edition can be used as a front-end web server or for departmental or branch office applications.

Workgroup Edition includes most of the core database features and capabilities of the SQL Server Standard Edition except for the following:

- It is limited to two CPUs and a maximum of 3GB of memory.
- It does not support failover clustering or database mirroring.
- Does not include Analysis Services or Notification Services.
- It provides limited support for Reporting Services features.

Developer Edition

The Developer Edition of SQL Server 2005 is a full-featured version intended for development and end-user testing only. It includes all the features and functionality of Enterprise Edition, at a much lower cost, but the licensing agreement prohibits production deployment of databases using Developer Edition.

To provide greater flexibility during development, Developer Edition can be installed in any of the following environments:

- Windows 2000 Professional
- Any Windows 2000 Server editions
- Any Windows 2003 Server editions
- Windows XP

Express Edition

The Express Edition of SQL Server 2005 is intended for users who are running applications that require a locally installed database, often on mobile systems, and who spend at least some time disconnected from the network. It replaces the Desktop Edition that was available with SQL Server 2000. Express Edition is a free, lightweight, embeddable and redistributable version of SQL Server 2005. SQL Server Express Edition includes a stripped-down version of SQL Server Management Studio, called SQL Server Management Studio Express, for easily managing a SQL Server Express instance and its databases. Best of all, as your needs grow, your applications seamlessly work with the rest of the SQL Server product family.

The Express Edition can be installed in any of the following environments:

- Windows 2000 Professional
- Any Windows 2000 Server edition
- Any Windows 2003 Server edition
- Windows XP

Express Edition supports most of the same features as the Workgroup Edition, with the following exceptions:

- It is limited to using a maximum of one CPU and 1GB of memory.
- It limits the maximum database size to 4GB.
- It does not include Full-Text Search, Notification Services, Reporting Services, or Analysis Services.
- It does not include SQL Server Integration Services.
- It supports Service Broker as a client only.
- It does not include SSMS.
- It can participate in replication only as a subscriber.

If you need a bit more than the Express Edition offers, but not as much as the Workgroup Edition, Microsoft also provides the Express Edition with Advanced Services. The Express Edition with Advanced Services includes support for Full-Text Search and limited support of Reporting Services for web reporting.

Mobile Edition

The Mobile Edition of SQL Server 2005 is a compact version of SQL Server 2005 that provides T-SQL compatibility and a cost-based Query Optimizer that runs on Windows Mobile devices, including devices running Microsoft Windows CE 5.0, Microsoft Windows XP Tablet PC Edition, Windows Mobile 2003 Software for Pocket PC, and Windows Mobile 5.0. Developers who are familiar with SQL Server 2005 should feel comfortable developing for Mobile Edition.

SQL Server Mobile Edition has a small footprint, requiring only about 2MB. SQL Server Mobile Edition can connect directly with a SQL Server 2005 database through remote execution of T-SQL statements and also supports replication with SQL Server 2005 databases as a merge replication subscriber so that data can be accessed and manipulated offline and synchronized later with server-based version of SQL Server 2005.

SQL Server Licensing Models

In addition to feature sets, one of the determining factors in choosing a SQL Server edition is cost. With SQL Server 2005, Microsoft provides two types of licensing models: processor-based licensing and server-based licensing.

Processor-based licensing requires a single license for each physical CPU in the machine that is running a Microsoft Server product. This type of license includes unlimited client device access. Additional server licenses, seat licenses, and Internet connector licenses are not required. You must purchase a processor license for each installed processor on the server on which SQL Server 2005 will be installed, even if some processors will not be

used for running SQL Server. The only exception is for systems with 16 or more processors that allow partitioning of the processors into groups so the SQL Server software can be delegated to a subset of the processors.

NOTE

For licensing purposes, Microsoft bases the number of CPUs in a machine on the number of CPU sockets on the motherboard, not the number of cores on the CPU chip itself. Thus, although a dual-core or quad-core processor chip may appear to the operating system as two or four CPUs, at the time of this writing, each these types of chips is still considered a single CPU for licensing purposes if it occupies only a single CPU socket on the motherboard.

For those who prefer the more familiar server/client access license (CAL), or for environments in which the number of client devices connecting to SQL Server is small and known, two server/CAL-based licensing models are also available:

- **Device CALs** —A device CAL is required in order for a device (for example, PC, workstation, terminal, PDA, mobile phone) to access or use the services or functionality of Microsoft SQL Server. The server plus device CAL model is likely to be the more cost-effective choice if there are multiple users per device (for example, in a call center).
- **User CALs**—A SQL server user CAL is required in order for a user (for example, an employee, a customer, a partner) to access or use the services or functionality of Microsoft SQL Server. The server plus user CAL model is likely to be more cost-effective if there are multiple devices per user (for example, a user who has a desktop PC, laptop, PDA, and so forth).

The server/CAL licensing model requires purchasing a license for the computer running SQL Server 2005 as well as a license for each client device or user that accesses any SQL Server 2005 installation. A fixed number of CALs are included with a server license and the server software. Additional CALs can be purchased as needed.

Server/per-seat CAL licensing is intended for environments in which the number of clients per server is relatively low, and access from outside the firewall is not required. Be aware that using a middle-tier or transaction server that pools or multiplexes database connections does not reduce the number of CALs required. A CAL is still required for each distinct client workstation that connects through the middle tier. (Processor licensing might be preferable in these environments due to its simplicity and affordability when the number of clients is unknown and potentially large.)

The pricing listed in Table 1.2 is provided for illustrative purposes only and is based on pricing available at the time of publication. These are estimated retail prices that are subject to change and might vary from reseller pricing.

TABLE 1.2 SQL Server 2005 Estimated Retail Pricing

Licensing Option	Enterprise Edition	Standard Edition	Workgroup Edition
Processor Licensing	$24,999 per processor	$5,999 per processor	$3,899 per processor
Server/per-seat CAL license with 5 workgroup CALs	N/A	N/A	$739
Server/per-seat CAL license with 5 CALs	N/A	$1,849	N/A
Server/per-seat CAL license with 25 CALs	$13,969	N/A	N/A

Developer Edition Licensing

The Developer Edition of SQL Server 2005 is available for a fixed price of $49.95. The Developer Edition is licensed per developer and must be used for designing, developing, and testing purposes only.

Express Edition Licensing

The Express Edition of SQL Server 2005 is available via free download from www.microsoft.com/sql. Developers can redistribute it with their applications at no cost by simply registering for redistribution rights with Microsoft. The Express Edition does not require a CAL when it is used on a standalone basis. If it connects to a SQL Server instance running Enterprise Edition, Standard Edition, or Workgroup Edition, a separate user or device CAL is required unless the SQL Server instance it connects to is licensed under the per-processor model.

Mobile Edition Licensing

SQL Server 2005 Mobile Edition is available as a downloadable development product for mobile applications. You can deploy SQL Server Mobile to an unlimited number of mobile devices if they operate in standalone mode (that is, the device does not connect to or use the resources of any SQL Server system not present on the device). If the device connects to a SQL Server instance that is not on the device, a separate user or device CAL is required unless the SQL Server instance is licensed under the per-processor model.

Choosing a Licensing Model

Which licensing model should you choose? Per-processor licensing is generally recommended for instances in which the server will be accessed from the outside. This includes servers used in Internet situations or servers that will be accessed from both inside and outside an organization's firewall. Per-processor licensing might also be appropriate and cost-effective for internal environments in which there are a very large number of users in relation to the number of SQL Server machines. An additional advantage to the per-processor model is that it eliminates the need to count the number of devices connecting

to SQL Server, which can be difficult to manage on an ongoing basis for a large organization.

Using the server/per-seat CAL model is usually the most cost-effective choice in internal environments in which client-to-server ratios are low.

Mixing Licensing Models

You can mix both per-processor and server/CAL licensing models in your organization. If the Internet servers for your organization are segregated from the servers used to support internal applications, you can choose to use processor licensing for the Internet servers and server/CAL licensing for internal SQL Server instances and user devices.

Keep in mind that you do not need to purchase CALs to allow internal users to access a server already licensed via a processor license: The processor licenses allow access to that server for all users.

Passive Server/Failover Licensing

In SQL Server 2005, two or more servers can be configured in a failover mode, with one server running as a passive server so that the passive server picks up the processing of the active server only in the event of a server failure. SQL Server 2005 offers three types of failover support:

- Database mirroring
- Failover clustering
- Log shipping

If your environment uses an active/passive configuration in which at least one server in the failover configuration does not regularly process information but simply waits to pick up the workload when an active server fails, no additional licenses are required for the passive server. The exception is if the failover cluster is licensed using the per-processor licensing model and the number of processors on the passive server exceeds the number of processors on the active server. In this case, additional processor licenses must be acquired for the number of additional processors on the passive computer.

In an active/active failover configuration, all servers in the failover configuration regularly process information independently unless a server fails, at which point one server or more takes on the additional workload of the failed server. In this environment, all servers must be fully licensed using either per-processor licensing or server/CAL licensing. Keep in mind that in some log shipping and database mirroring configurations, the standby (passive) server can be used as a read-only reporting server installation. Under this usage, the standby server is no longer “passive” and must be licensed accordingly.

Virtual Server Licensing

Virtualization is defined broadly as the running of software on a "virtual environment." A virtual environment exists when an operating system is somehow emulated (that is, does not run directly on the physical hardware). When you're running virtualization software on a system, one or several applications and their associated operating systems can run on one physical server inside their respective virtual environments.

Running SQL Server 2005 inside a virtual operating environment requires at least one license per virtual operating environment. Within each virtual operating environment, the license allows you to run one or more instances of SQL Server 2005. The license for a virtual operating environment can be a server/CAL license or a processor-based license. If using a processor based license, you must purchase a processor license for each processor that the virtual machine accesses.

Summary

This chapter examines the various platforms that support SQL Server 2005 and reviews and compares the various editions of SQL Server 2005 that are available. Which platform and edition are appropriate to your needs depends on scalability, availability, performance, licensing costs, and limitations. The information provided in this chapter should help you make the appropriate choice.

Chapter 2, "What's New in SQL Server 2005," takes at closer look at the new features and capabilities provided with the various SQL Server 2005 editions.

This page intentionally left blank

CHAPTER 2

What's New in SQL Server 2005

IN THIS CHAPTER

- New SQL Server 2005 Features
- SQL Server 2005 Enhancements

The upgrade from SQL Server 6.5 to 7.0 was pretty significant. In addition to many new features, the underlying SQL Server architecture changed considerably. In comparison, the upgrade from SQL Server 7.0 to 2000 was more of a series of enhancements, additions, and improvements.

Microsoft SQL Server 2005 further extends the performance, reliability, availability, programmability, and ease of use of SQL Server 2000. SQL Server 2005 includes several new features that make it an excellent database platform for large-scale online transaction processing (OLTP), data warehousing, and e-commerce applications.

This chapter explores the new features provided in SQL Server 2005 as well as many of the enhancements to previously available features.

New SQL Server 2005 Features

What does SQL Server 2005 have to offer over SQL Server 2000? The following is a list of the new features provided in SQL Server 2005:

- SQL Server Management Studio (SSMS)
- SQL Server Configuration Manager
- Common language runtime (CLR)/.NET Framework integration
- Dynamic management views (DMVs)
- System catalog views
- SQL Server Management Objects (SMO)
- Dedicated administrator connection (DAC)

- SQLCMD
- Database Mail
- Online index and restore operations
- Native encryption
- Database mirroring
- Database snapshots
- Service Broker
- SQL Server Integration Services (SSIS)
- Table and index partitioning
- Snapshot isolation
- Business Intelligence (BI) Development Studio
- Query Notification
- Multiple active result sets
- New SQL Server data types

The rest of this section takes a closer look at each of these new features and, where appropriate, provides references to subsequent chapters where you can find more information about the new features.

SQL Server Management Studio

One of the biggest changes in SQL Server 2005 that you will notice right away if you have experience with SQL Server 2000 is that SQL Enterprise Manager and Query Analyzer are no longer provided with SQL Server 2005. They have been replaced by a single integrated management console called SQL Server Management Studio (SSMS). SSMS is the tool to use to monitor and manage your SQL Server database engines and databases, as well as Integration Services, Analysis Services, Reporting Services, and Notification Services across your entire SQL Server enterprise. Improvements to the SSMS interface over SQL Enterprise Manager allows database administrators to perform several tasks at the same time, such as authoring and executing a query, viewing server objects, managing objects, monitoring system activity, and viewing online help.

SSMS also provides a Visual Studio–like development environment for authoring, editing, and managing scripts and stored procedures, using Transact-SQL (T-SQL), Multidimensional Expressions (MDX), XML for Analysis, and SQL Server Mobile Edition. SSMS can also integrate with source control software such as Visual SourceSafe to allow you to define and manage your scripts under projects.

SSMS also hosts tools for scheduling SQL Server Agent jobs and managing maintenance plans to automate daily maintenance and operation tasks. The integration of management and authoring in a single tool, coupled with the ability to manage all types of servers, provides enhanced productivity for database administrators.

For an introduction to the capabilities and how to make the most of SSMS, see Chapter 3, "SQL Server Management Studio." In addition, additional features of SSMS are shown in more detail throughout this entire book.

SQL Server Configuration Manager

SQL Server 2005 introduces SQL Server Configuration Manager, a management tool that allows administrators to manage SQL Server services and configure basic service and network protocol options. SQL Server Configuration Manager combines the functionality of the following SQL Server 2000 tools:

- Server Network Utility
- Client Network Utility
- Services Manager

SQL Server Configuration Manager also includes the ability to start and stop and set service properties for the following services:

- SQL Server
- SQL Server Agent
- SQL Server Analysis Services (SSAS)
- Report server
- Microsoft Distributed Transaction Coordinator (MS DTC)
- Full-Text Search

The use of SQL Server Configuration Manager is discussed in various chapters of this book where its use is appropriate.

CLR/.NET Framework Integration

In SQL Server 2005, database programmers can now take full advantage of the Microsoft .NET Framework class library and modern programming languages to implement within SQL Server itself stored procedures, triggers, and user-defined functions written in the .NET Framework language of their choice. Many tasks that were awkward or difficult to perform in T-SQL can be better accomplished by using managed code.

By using languages such as Visual Basic .NET and C#, you can capitalize on CLR integration to write code that has more complex logic and is more suited for computational

tasks, such as string manipulation or complex mathematical calculation. Managed code is more efficient than T-SQL at processing numbers and managing complicated execution logic, and it provides extensive support for string handling, regular expressions, and so on.

In addition, two new types of database objects—aggregates and user-defined types—can be defined using the .NET Framework. With user-defined types, you can define your own type that can be used for column definitions (for example, custom date/time data types, currency data types), or you can define other complex types that may contain multiple elements and can have behaviors, differentiating them from the traditional SQL Server system data types.

In addition, you may at times need to perform aggregations over data, such as statistical calculations. If the desired aggregation function is not directly supported as a built-in aggregate function, you have the option to define a custom user-defined aggregate, using the .NET Framework to perform a custom aggregation in SQL Server 2005.

For more information on CLR/.NET Framework integration in SQL Server 2005, see Chapter 36, "SQL Server and the .NET Framework."

Dynamic Management Views

Dynamic Management Views (DMVs) are new to SQL Server 2005 and provide a light-weight means for accessing information about internal database performance and resource usage without the heavy burden associated with tools used in SQL Server 2000. DMVs provide information on internal database performance and resource usage, ranging from memory, locking, and scheduling to transactions and network and disk I/O. DMVs can be used to monitor the health of a server instance, diagnose problems, and tune performance.

An extensive number of DMVs are available in SQL Server 2005. Some DMVs are scoped at the server level, and others are scoped at the database level. They are all found in the `sys` schema and have names that start with `dm_`. See Chapter 6, "SQL Server System and Database Administration," for more information on the DMVs available in SQL Server 2005 and how to use them. In addition, other more detailed examples of using DMVs are provided in the chapters in the SQL Server Performance and Optimization section of this book.

System Catalog Views

In SQL Server 2005, the familiar system catalog tables have been replaced with system catalog views. Using catalog views is the preferred method for viewing information that is used by the Microsoft SQL Server database engine. There is a catalog view to return information about almost every aspect of SQL Server. Some of the catalog views return information that is new to SQL Server 2005 or information that was not provided in prior versions. Examples of these include the CLR assembly catalog views and the database mirroring catalog views. Other catalog views provide information that may have been available in prior versions via system tables, system procedures, and so on, but the new

catalog views expand on the information that is returned and include elements that are new to SQL Server 2005.

SQL Server 2005 stores the system catalogs in a new hidden system database called the *resource database*. Microsoft chose to make the resource database inaccessible to ensure quick, clean upgrades and to allow rollbacks of intermediate updates or bug fix releases. By implementing the system catalog as a set of views rather than as a directly accessible base table gives Microsoft the flexibility to adjust the catalog schema in the future without affecting existing applications.

For more detailed information on the system catalog views available in SQL Server 2005, see Chapter 6.

SQL Server Management Objects

SQL Server 2005 provides a new programmatic access layer called SQL Server Management Objects (SMO), which is a new set of programming objects that exposes all the management functionality of the SQL Server database. SMO replaces Distributed Management Objects (DMO), which was included with earlier versions of SQL Server. SMO provides improved scalability and performance over DMO.

Dedicated Administrator Connection

In previous releases of SQL Server, under certain circumstances, the system could become inaccessible due to running out of available user connections or other resources that could prevent normal access of the server, even by a system administrator. To help solve this problem, SQL Server 2005 introduces the Dedicated Administrator Connection (DAC), which allows an administrator to access a running server even if the server is not responding or is otherwise unavailable. Through the DAC, the administrator can execute diagnostic functions or T-SQL statements to troubleshoot problems on a server.

SQL Server listens for the DAC on a dedicated TCP/IP port dynamically assigned upon database engine startup. The error log contains the port number the DAC is listening on. By default, the DAC listener accepts connection on only the local port via SSMS or the `sqlcmd` command-prompt tool. Only members of the SQL Server `sysadmin` role can connect using the DAC.

To connect to the DAC using the `sqlcmd` command-prompt utility, you specify the special administrator switch (`-A`). To connect to the DAC via SSMS, you prefix `admin:` to the instance name in the connection dialog.

SQLCMD

SQLCMD is the next evolution of the `isql` and `osql` command-line utilities that you may have used in prior versions of SQL Server. It provides the same type of functionality as `isql` or `osql`, including the ability to connect to SQL Server from the command prompt and execute T-SQL commands. It also offers a number of new script execution options that go beyond what was available before. You can run queries interactively in SQLCMD

(as you can with `isql` and `osql`), but the real power of SQLCMD comes into play when you use it to automate T-SQL scripts that are invoked by batch files.

SQLCMD includes a number of new internal and external commands and scripting variables that enhance the execution of T-SQL. With scripting variables, you can store values in variables and have the values span batches.

For more information on using SQLCMD, see Chapter 4, "SQL Server Command-Line Utilities."

Database Mail

SQL Server 2005 introduces Database Mail as a replacement for SQLMail. Database Mail includes a number of new features and enhancements over SQLMail, including support for multiple email profiles and accounts, asynchronous (queued) message delivery via a dedicated process in conjunction with Service Broker, cluster-awareness, 64-bit compatibility, greater security options (such as controlling the size of mail attachments and prohibiting specified file extensions), and simpler mail auditing. Database Mail also utilizes industry-standard Simple Mail Transfer Protocol (SMTP), so you no longer have to have an Extended MAPI mail client installed on the SQL Server machine.

For more information on configuring and using Database Mail, see Chapter 12, "Database Mail."

Online Index and Restore Operations

A new feature of SQL Server 2005 Enterprise Edition is the ability to create, rebuild, or drop an index online. The online index option allows concurrent modifications (updates, deletes, and inserts) to the underlying table or clustered index data and any associated indexes while the index operations are ongoing. This feature allows you to add indexes without interfering with access to tables or other existing indexes.

SQL Server 2005 Enterprise Edition also introduces the ability to perform a restore operation while the database is online. During this operation, only the data that is being restored is unavailable. The rest of the database remains online and available. Earlier versions of SQL Server required that the entire database be offline during the restore process.

For more information on online index operations, see Chapter 20, "Creating and Managing Indexes." For information on online restore, see Chapter 11, "Database Backup and Restore."

Native Encryption

While some encryption functionality existed in previous versions of SQL Server (for example, involving column encryption APIs within user-defined functions or the `PWDENCRYPT` password one-way hash function), it was relatively limited and rarely used. SQL Server 2005 provides significant improvements in this area. SQL Server 2005 introduces built-in native encryption capabilities, which includes added encryption tools, certificate creation, and key management functionality that can be invoked within T-SQL.

T-SQL now includes support for symmetric encryption and asymmetric encryption using keys, certificates, and passwords. SQL Server 2005 includes several new functions to securely create, manage, and use encryption keys and certificates to secure sensitive data. Taking advantage of this new functionality can greatly enhance your database and application security.

Database Mirroring

In SQL Server 2000, there were only two viable methods for maintaining a warm-standby copy of a database: replication and log shipping. SQL Server 2005 introduces another method of maintaining a hot- or warm-standby database: database mirroring. Database mirroring provides the continuous streaming of the transaction log from a source server to a single destination server. In the event of a failure of the primary system, applications can immediately reconnect to the database on the secondary server. The secondary instance detects failure of the primary server within seconds and accepts database connections immediately. Database mirroring works on standard server hardware and requires no special storage devices or controllers.

For more information on configuring and using database mirroring, see Chapter 16, "Database Mirroring."

NOTE

Database mirroring was not available in the GA release of SQL Server 2005 but is included with Service Pack 1.

Database Snapshots

SQL Server 2005 enables database administrators to create database snapshots, which are instant, read-only views of a database that capture the state of the database at a specific point in time. A database snapshot can be used to provide a stable view of data without incurring the time or storage overhead of creating a complete copy of the database. As data in the primary database is modified after the snapshot has been taken, the snapshot facility makes a copy of the data page(s) being modified so it has a pre-modified copy of the page(s), which provide a point-in-time view of the data. In addition to providing a point-in-time snapshot of a database for reporting purposes (for example, an end-of-month snapshot of a database), a snapshot can also be used to quickly recover from an accidental change to a database. The original pages from the snapshot can be quickly copied back to the primary database.

For more information on creating and using database snapshots, see Chapter 27, "Database Snapshots."

Service Broker

The Service Broker feature of SQL Server 2005 provides a scalable architecture for building asynchronous message routing. The Service Broker technology allows internal or external

processes to send and receive streams of reliable, asynchronous messages by using extensions to normal T-SQL. Messages can be sent to a queue in the same database as the sender, to another database in the same instance of SQL Server, or to another instance of SQL Server, either on the same server or on a remote server. Service Broker handles all the hard work of managing coordination among the constructs required for distributed messaging, including transactional delivery and storage, message typing and validation, multithreaded activation and control, event notification, routing, and security.

With Service Broker, SQL Server 2005 provides the ability to build loosely coupled, asynchronous database applications. For example, in an order entry system, the system needs to process some parts of an order, such as product availability and pricing, before the order is considered complete. However, other parts of the order, such as billing and shipping, don't have to happen before the system commits the order. If a system can process the parts of the order that can be delayed in a guaranteed but asynchronous manner, an organization can process the core part of the order faster. Service Broker provides this capability.

For more information on Service Broker, see Chapter 48, "SQL Server Service Broker" (on the CD-ROM).

SQL Server Integration Services

SQL Server 2005 replaces Data Transformation Services (DTS) with a completely redesigned enterprise data extraction, transformation, and loading (ETL) platform called SQL Server Integration Services (SSIS). SSIS provides a number of enhancements over what was available in DTS, including the following:

- A revamped GUI interface hosted in a Visual Studio shell for building, debugging, and deploying SSIS packages
- SSIS application programming interfaces (APIs) for developing custom SSIS components
- New control flow components, such as `ForEach Loop`, `For Loop`, and `Sequence`
- A service-based implementation (the Integration Services service)

For more information on designing and using SSIS, see Chapter 40, "SQL Server Integration Services."

Table and Index Partitioning

In SQL Server 2005 tables and indexes are stored in one or more partitions. Partitions are organizational units that allow you to divide your data into logical groups. Table and index partitioning eases the management of large databases by facilitating the management of a database in smaller, more manageable chunks. For example, a date/time column can be used to divide each month's data into a separate partition. You can assign partitions to different filegroups for added flexibility and ease of maintenance.

Tables with multiple partitions (that is, partitioned tables) are accessed the same way as single-partition tables. DML operations such as `INSERT` and `SELECT` statements reference the table the same way, regardless of partitioning.

Generally, partitioning is most useful for large tables. *Large* is a relative term, but these tables typically contain millions of rows and take up gigabytes of space. Oftentimes, the tables that are targeted for partitioning are large tables that are experiencing performance problems because of their size.

Partitioning provides advantages for many different scenarios, including the following:

- **Archival and purging**—Table partitions can be quickly switched from a production table to another archive table with the same structure, allowing you to keep a limited amount of recent data in the production table while keeping the bulk of the older data in the archive table. Alternatively, the data in a partition can be switched out to a staging table that can then either be archived or truncated to purge the data.
- **Maintenance**—Table partitions that have been assigned to different filegroups can be backed up and maintained independently of each other.
- **Query performance**—Partitioned tables that are joined on partitioned columns can experience improved performance because the Query Optimizer can join to the tables based on the partitioned column. Queries can also be parallelized along the partitions.

For more information on table partitioning, see Chapter 19, "Creating and Managing Tables."

Snapshot Isolation

One of the key new features of SQL Server 2005 related to locking and performance is snapshot isolation. Snapshot isolation provides the benefit of repeatable reads without the need to acquire and hold shared locks on the data that is read. The snapshot isolation level allows users to access the last row that was committed by using a transactionally consistent view of the data.

This new isolation level provides the following benefits:

- Increased data availability for read-only applications
- Minimized locking and blocking problems between read operations and update operations in an OLTP environment
- Automatic mandatory conflict detection for write transactions

For more information on snapshot isolation, see Chapter 32, "Locking and Performance."

Business Intelligence Development Studio

SQL Server 2005 introduces the Business Intelligence Development Studio, which is a Visual Studio–based development environment for building business intelligence (BI) solutions that includes templates and project types that are specific to SQL Server 2005 business intelligence. It provides a solutions-based approach for developing BI solutions that includes Analysis Services, Integration Services, and Reporting Services projects.

For more information and a tutorial on designing Analysis Services solutions using Business Intelligence Development Studio, see Chapter 39, "SQL Server 2005 Analysis Services."

Query Notification

With the availability of the Service Broker, SQL Server 2005 also introduces notification support for SQL Server queries. Query Notification is useful for applications that cache database query results in their own private cache area, such as database-driven websites. Rather than having the application repeatedly poll the database to determine whether the data has changed and the cache needs to be refreshed, commands that are sent to the server through any of the client APIs, such as ADO.NET, OLE DB, open database connectivity (ODBC), Microsoft ActiveX Data Objects (ADO), or Simple Object Access Protocol (SOAP), may include a tag that requires a notification. For each query included in the request, SQL Server creates a notification subscription. When the data changes, a notification is delivered through a SQL Service Broker queue to notify the application that data has changed, at which point the application can refresh its data cache.

For more information on using Query Notification, see Chapter 36.

Multiple Active Result Sets

In previous versions of SQL Server, a user connection could have only one pending request per user connection. When using SQL Server default result sets, the application had to process or cancel all result sets from one batch before it could execute any other batch on that connection. SQL Server 2005 provides multiple active result sets (MARS), which allows you to have more than one default result set open per connection simultaneously. In other words, applications can have multiple default result sets open and can interleave reading from them or applications can execute other statements (for example, `INSERT`, `UPDATE`, `DELETE`, and stored procedure calls) while default result sets are open. The existing result set does not need to be cancelled or fully processed first.

For more information on using MARS and configuring client connections to enable MARS, see Chapter 36 and Chapter 9, "Client Installation and Configuration."

New SQL Server Data Types

SQL Server 2005 introduces a brand-new data type, `xml`, and a new size specification, `max`, for `varchar` and `varbinary` data types.

The `xml` Data Type

One of the biggest limitations of XML in SQL Server 2000 was the inability to save the results of a `FOR XML` query to a variable or store it in a column directly without using some middleware code to first save the XML as a string and then insert it into an `ntext` or `nvarchar` column and then select it out again. SQL Server 2005 now natively supports column storage of XML, using the new `xml` data type.

The new `xml` data type allows relational columns and XML data to be stored side-by-side in the same table. Some of the benefits of storing XML in the database include the traditional DBMS benefits of backup and restore, replication and failover, query optimization, granular locking, indexing, and content validation. In SQL Server 2005, an XML index can be created on both untyped and typed XML columns, and it indexes all paths and values within the entire XML column for faster searching and retrieval of XML data.

The `xml` data type can also be used with local variable declarations, as the output of user-defined functions, as an input parameter to stored procedures and functions, and much more. XML columns can also be used to store code files, such as XSLT, XSD, XHTML, and any other well-formed content. These files can then be retrieved by user-defined functions written in managed code hosted by SQL Server.

For more information on using the `xml` data type in SQL Server 2005, see Chapter 37, "Using XML in SQL Server 2005."

`varchar(max)` and `varbinary(max)`

In SQL Server 2000, the largest value that could be stored in a `varchar` or `varbinary` column was 8000 bytes. If you needed to store a larger value in a single column, you had to use the `text` or `image` data type. The main disadvantage of using the `text` and `image` data types was that they could not be used in many places where `varchar` or `varbinary` data types could be used (for example, as arguments to SQL Server's string manipulation functions, such as `SUBSTRING`, `CHARINDEX`, and `REPLACE`).

SQL Server 2005 introduces the new `max` specifier for `varchar` and `varbinary` data types. This specifier expands the storage capabilities of the `varchar` and `varbinary` data types to store up to 2^{31}–1 bytes of data, the same maximum size as text and image data types. The main difference is that the new `max` data types can be used just like regular `varchar` and `varbinary` data types in functions, for comparisons, as T-SQL variables, and for concatenation. They can also be used in the `DISTINCT`, `ORDER BY`, and `GROUP BY` clauses of a `SELECT` statement, as well as in aggregates, joins, and subqueries.

For more information on using the `max` data types, see Chapter 35, "What's New for Transact-SQL in SQL Server 2005."

SQL Server 2005 Enhancements

In addition to the new features in SQL Server 2005, there are a number of enhancements to existing features. This section provides an overview of the major enhancements provided in SQL Server 2005.

Database Engine Enhancements

Several new database-specific enhancements have been added to SQL Server 2005. These changes are focused primarily on the database storage engine. The following are some of the most important enhancements:

- **Instant file initialization**—New or expanded database files are made available much faster now because the initialization of the file with binary zeros is deferred until data is written to the files.
- **Partial availability**—In the event of database file corruption, the database can still be brought online if the primary filegroup is available.
- **Database file movement**—You can now use the `ALTER DATABASE` command to move a database file. The physical file must be moved manually. This feature was available in SQL Server 2000, but it only worked on `tempdb`.

In addition, many new table-oriented enhancements are available with SQL Server 2005. This includes features that define how the data in the tables will be stored in the database. The following are two of the key enhancements:

- **Large rows**—SQL Server 2005 now allows for the storage of rows that are greater than 8060 bytes. The 8060-byte limitation that existed with SQL Server 2000 has been relaxed by allowing the storage of certain data types (such as `varchar` and `nvarchar`) on a row overflow data page.
- **Stored computed columns**—Computed columns that were calculated on-the-fly in prior versions can now be stored in the table structure. You accomplish this by specifying the `PERSISTED` keyword as part of the computed column definition.

Index Enhancements

The following are some of the most important enhancements available for indexes with SQL Server 2005:

- **Included columns**—Non-key columns can now be added to an index for improved performance. The performance gains are achieved with covering indexes that allow the Query Optimizer to locate all the column values referenced in the query.
- **`ALTER INDEX`**—As with other database objects, such as tables and databases, you can now modify indexes by using the `ALTER` statement. Index operations that were previously performed with DBCC commands or system stored procedures can now be accomplished with the `ALTER INDEX` command.
- **Parallel index operations**—Scan and sort activities associated with index operations can now be done in parallel.

For more information on making use of the enhanced index features, see Chapter 20 and Chapter 29, "Indexes and Performance."

T-SQL Enhancements

SQL Server 2005 provides many enhancements to the T-SQL language that allow you to improve the performance of your code and extend your error-management capabilities. These enhancements include improved error handling, new recursive query capabilities, and support for new SQL Server database engine capabilities. Some of the T-SQL enhancements are as follows:

- **Ranking functions**—SQL Server 2005 introduces four new ranking functions: `ROW_NUMBER`, `RANK`, `DENSE_RANK`, and `NTILE`. These new functions allow you to efficiently analyze data and provide ranking values to result rows of a query.
- **Common table expressions**—A common table expression (CTE) is a temporary named result set that can be referred to within a query, similarly to a temporary table. CTEs can be thought of as an improved version of derived tables that more closely resemble a non-persistent type of view. You can also use CTEs to develop recursive queries that you can use to expand a hierarchy.
- **`PIVOT`/`UNPIVOT` operator**—The `PIVOT` operator allows you to generate crosstab reports for open-schema and other scenarios in which you rotate rows into columns, possibly calculating aggregations along the way and presenting the data in a useful form. The `UNPIVOT` operator allows you to normalize pre-pivoted data.
- **`APPLY`**—The `APPLY` relational operator allows you to invoke a specified table-valued function once per each row of an outer table expression.
- **`TOP` enhancements**—In SQL Server 2005, the `TOP` operator has been enhanced, and it now allows you to specify a numeric expression to return the number or percentage of rows to be affected by your query; you can optionally use variables and subqueries. You can also now use the `TOP` option in `DELETE`, `UPDATE`, and `INSERT` queries.
- **DML with results**—SQL Server 2005 introduces a new `OUTPUT` clause that allows you to return data from a modification statement (`INSERT`, `UPDATE`, or `DELETE`) to the processing application or into a table or table variable.
- **Exception handling for transactions—**Earlier versions of SQL Server required you to include error-handling code after every statement that you suspected could potentially generate an error. SQL Server 2005 addresses this by introducing a simple but powerful exception-handling mechanism in the form of a `TRY...CATCH` T-SQL construct.

For more information on the new and enhanced features of T-SQL, see Chapter 35.

Security Enhancements

SQL Server 2005 includes significant enhancements to the security model of the database platform, with the intention of providing more precise and flexible control to enable

tighter security of data. Some of the new features and enhancements to improve the level of security for your enterprise data include the following:

- **SQL login password policies**—SQL Server logins can now be governed by a more rigid password policy. This is implemented with new `CHECK_POLICY` and `CHECK_EXPIRATION` options that can be selected for a SQL Server login. These options facilitate stronger passwords and cause passwords to expire. The password policy is enforced only on Windows 2003 Server and above.
- **User/schema separation**—In prior versions of SQL Server, the fully qualified name for every object was directly tied to the object owner. With SQL Server 2005, schema names are used in the object namespace instead. This user/schema separation provides more flexibility in the object model and allows for object owners to be changed without affecting the code that references the objects.
- **Module execution context**—The `EXECUTE AS` option can be used to set the execution context for SQL statements. This allows a user to impersonate another user and is particularly useful for testing permissions.
- **Permission granularity**—The security model in SQL Server 2005 provides a much more granular level of control than earlier versions of SQL Server. This granular control provides some new types of security and allows you to apply security to a new set of database objects.
- **Data Definition Language (DDL) triggers**—Database administrators can now write server- or database-level triggers that fire on events such as creating or dropping a table, adding a server login, or altering a database. These triggers provide an invaluable auditing mechanism to automatically capture these events as they occur.

For more information on the security features in SQL Server 2005, see Chapter 10, "Security and User Administration." For more information on DDL triggers, see Chapter 25, "Creating and Managing Triggers."

Backup and Restore Enhancements

SQL Server builds on the basic set of backup and restore features that exist in SQL Server 2000 with enhancements such as the following:

- **Copy-only backups**—These backups can be made without disrupting the sequencing of other backups (for example, differential or log backups). Because copy-only backups do not affect the restoration chain, they are useful in situations such as when you simply want to get a copy of the database for testing purposes.
- **Mirrored backups**—SQL Server 2005 adds the capability to create additional copies of database backups via mirrored backups. Mirrored backups provide redundancy so that you can overcome the failure of a single backup device or medium by utilizing the mirrored copy of the backup.

- **Partial backups**—A partial backup contains all the data in the primary filegroup, any filegroup that is not read-only, and any filegroup that has been explicitly identified for backup. The elimination of read-only filegroups from partial backups saves space, saves time, and reduces the server overhead that is required while performing the backup.
- **Online restore**—Online restore allows a filegroup, file, or a specific page within a file to be restored while the rest of the database remains online and available.

See Chapter 11 for more information on the backup and restore capabilities of SQL Server 2005.

SQL Server Agent Enhancements

Microsoft has continued to improve the capabilities of the SQL Server Agent. It has maintained a consistent basis for automation while enriching the feature set. The following are some of the key new features:

- **Job Activity Monitor**—A new auto-refreshing tool named Job Activity Monitor has been added to help monitor the execution of scheduled jobs. You can adjust the refresh rate of the screen and specify filtering criteria in order to isolate a job or set of jobs.
- **Shared job schedules**—A job schedule can now be shared among jobs that have the same job owner.
- **Enhanced SQL Agent security**—Several new roles have been added that provide enhanced security management for the SQL Server Agent. In addition, a separate proxy account can now be defined for each type of subsystem that the SQL Server Agent can interact with.
- **Performance improvements**—New thread pooling and reduced job execution delays have improved the performance of SQL Server Agent.

See Chapter 13, "SQL Server Scheduling and Notification," for additional information on SQL Server Agent.

Recovery Enhancements

SQL Server 2005 reduces the amount of time it takes for a database to become available with a new and faster recovery option. In SQL Server 2005, users can access a recovering database after the transaction log has been rolled forward. Earlier versions of SQL Server required users to wait until incomplete transactions had rolled back, even if the users did not need to access the affected parts of the database.

Replication Enhancements

SQL Server 2005 offers a significant number of new features and improvements to replication. Much of what's new for SQL Server data replication revolves around simplifying setup, administration, and monitoring of a data replication topology, including the addition of a new Replication Monitor for ease of use in managing complex data replication operations. Replication Monitor has an intuitive user interface and a wealth of data metrics.

The following are some of the key replication enhancements:

- **Replication security enhancements**—The replication security model has changed, allowing more control over the accounts under which replication agents run.
- **Simplification of the user interface**—Replication wizards and dialog boxes have been redesigned for SQL Server 2005 to simplify the setup of a replication topology. There are 40% fewer wizard dialogs, and scripting is now integrated into the wizards, providing improved capabilities to script replication setup during or after wizard execution.
- **Replication of schema changes**—A much broader range of schema changes can be replicated without the use of special stored procedures. DDL statements are issued at the publisher and are automatically propagated to all subscribers.
- **Peer-to-peer transactional replication**—The new peer-to-peer model allows replication between identical participants in the topology (a master/master or symmetric publisher concept).
- **Initialization of a transactional subscription from a backup**—Setting up replication between databases that initially contain large volumes of data can be time-consuming and require large amounts of storage. SQL Server 2005 provides a new publication option that allows any backup taken after the creation of a transactional publication to be restored at the subscriber, rather than using a snapshot to initialize the subscription.
- **Heterogeneous replication**—Enhancements have been made for publishing data from an Oracle database with transactional and snapshot replication. In addition, there is improved support for many non-SQL Server subscribers.
- **Replication mobility**—Merge replication provides the ability to replicate data over HTTPS, with the web synchronization option, which is useful for synchronizing data from mobile users over the Internet or synchronizing data between Microsoft SQL Server databases across a corporate firewall.
- **Replication scalability and performance enhancements**—Scalability and performance enhancements include significant improvements to the performance of filtered merge publications and the ability of the Distribution Agent in transaction replication to apply batches of changes, in parallel, to a subscriber.

For more information on using and managing replication with SQL Server 2005, see Chapter 15, "Replication."

Failover Clustering Enhancements

In SQL Server 2005, support for failover clustering has been extended to SQL Server Analysis Services, Notification Services, and SQL Server replication. In addition, the maximum number of cluster nodes has been increased to eight. Other key enhancements to clustering include the following:

- **Simpler Microsoft Cluster Service (MSCS) setup**—Installing MSCS has become very easy (with Windows 2003 and above). MSCS is a prerequisite for SQL Server Clustering.
- **Cleaner SQL Server Clustering installation wizard**—The much-improved wizard detects and handles most prerequisites and provides for a single point of installation for multiple SQL Server node configurations.
- **Increased instances per cluster**—Up to 50 SQL Server instances per cluster are now supported with SQL Server 2005 Enterprise Edition and up to 16 SQL Server instances per cluster for SQL Server Standard Edition.
- **Number of nodes in a cluster**—With Windows 2003 Enterprise Edition, you can now create up to eight nodes in a single cluster.

SQL Server 2005 also provides the ability to set up a two-node failover cluster, using SQL Server 2005 Standard Edition. In previous releases, clustering was available only for the Enterprise Edition.

For more information on clustering, see Chapter 17, "SQL Server Clustering."

Notification Services Enhancements

Notification Services was provided as a feature for SQL Server 2000 before SQL Server 2005 was released. SQL Server 2005 provides a number of enhancements to Notification Services, including the following:

- **Integration into SSMS**—Notification Services is now integrated into SSMS Object Explorer. Using Object Explorer, you can perform most `nscontrol` command prompt utility tasks, and you can start and stop instances of Notification Services.
- **Support for subscriber-defined conditions**—In SQL Server 2005, Notification Services supports a new type of actions, called condition actions, which allow subscribers to define their own query clauses over a predefined data set. Using condition actions allows subscribers to fully define their own subscriptions over the data set.
- **Database independence**—SQL Server Notification Services supports using existing databases for instance and application data.

- **New management API**—SQL Server Notification Services has a new management API, `Microsoft.SqlServer.Management.Nmo`, that can be used to develop Notification Services instances and applications and to manage those instances and applications.

For more information on the Notification Server architecture and configuring and using SQL Server 2005 Notification Services, see Chapter 47, "SQL Server Notification Services" (on the CD-ROM).

Full-Text Search Enhancements

SQL Server 2005 provides a number of enhancements to the Full-Text Search service to improve the manageability and performance of Full-Text Search. The following are some of the major enhancements:

- **Integrated backup and restoration for full-text catalogs**—In SQL Server 2005, full-text catalogs can be backed up and restored along with, or separate from, database data.
- **Full-text catalogs included in database attach and detach operations**—SQL Server 2005 preserves full-text catalogs when administrators perform database detach and attach operations.
- **Full-text indexing performance improvements**—SQL Server 2005 Full-Text Search includes a major upgrade of the Microsoft Search service to version 3.0, which provides massively improved full-text index population performance and provides one instance of the Microsoft Search service per instance of SQL Server.

For more information on using Full-Text Search for SQL Server 2005, see Chapter 49, "SQL Server Full-Text Search" (on the CD-ROM).

Web Services Enhancements

The main enhancement to Web services in SQL Server 2005 is that you can use HTTP to access SQL Server directly, without using a middle-tier listener such as Microsoft Internet Information Services (IIS). SQL Server 2005 exposes a Web service interface to allow the execution of SQL statements and invocation of functions and procedures directly. Query results are returned in XML format and can take advantage of the Visual Studio Web services infrastructure.

For more information on using SQL Server Web Services in SQL Server 2005, see Chapter 38, "SQL Server Web Services."

Analysis Services Enhancements

SQL Server Analysis Services introduces a number of enhancements, including new management tools, an integrated development environment, and integration with the

.NET Framework. So many things have been changed that it's difficult to list them all. Here are some of the highlights of what has changed for Analysis Services:

- Analysis Services is now fully integrated with the SSMS. Many of the same wizards and management aspects of Business Intelligence Development Studio are also available in SSMS.
- SQL Server 2005 allows up to 50 separate instances of Analysis Services on one machine with Enterprise Edition and up to 16 separate instances with the Developer and Standard Editions.
- Analysis Services is now a cluster-aware application, and failover clustering is completely supported.
- SQL Server 2005 supports the XML for Analysis Services 1.1 specification and Analysis Services Scripting Language (ASSL) for XML-based administration.
- Proactive caching has been enabled at the partition level, to push data that has changed into cache for immediate access in Analysis Services. This is a big architectural change that directly addresses high-performance query execution of data within online analytical processing (OLAP) cubes that change frequently.
- The Unified Dimensional Model (UDM) paradigm provides a powerful metadata abstraction layer to use for all Analysis Services reference needs. It leverages concepts such as dimensions, measures, hierarchies, and so on and provides these simplified reference points to all interfaces and environments.
- Perspectives are now available to simplify and control the end user's view into complex cubes.
- Several new data mining algorithms have appeared, such as Naïve Bayes, Association, Sequence Clustering, Time Series/Linear Regression, and Neural Network algorithms.
- Analysis Services includes more robust usage and integration with SSIS for complex data transformations and filtering of data mining.

For more information on the new features and capabilities of SQL Server Analysis Services (SSAS), see Chapter 39.

Reporting Services Enhancements

The first version of SQL Server Reporting Services shipped in January 2004 for use with SQL Server 2000. However, Reporting Services is now an integrated component of SQL Server 2005, and the new version included in SQL Server 2005 contains a great deal of new features geared toward ease of use, performance, and the improvement of a rich development platform.

Reporting Services is a SQL Server service, similar to the relational database engine or Analysis Services. It allows you to design reports, deploy them on a server, and make them available to users in a secured environment in a variety of online and offline formats.

For more information on the capabilities of Reporting Services and how to use it, see Chapter 41, "SQL Server 2005 Reporting Services."

Summary

SQL Server 2005 provides a number of new and long-awaited features and enhancements. This chapter provides an overview of the new features and enhancements that ship with SQL Server 2005 and Service Pack 1 and Service Pack 2. To learn more, please refer to the other chapters referenced here.

PART II

SQL Server Tools and Utilities

IN THIS PART

This page intentionally left blank

CHAPTER 3

SQL Server Management Studio

IN THIS CHAPTER

- What's New in SSMS
- The Integrated Environment
- Administration Tools
- Development Tools

SQL Server Management Studio (SSMS) is a new integrated application that provides access to most of the graphical tools you can use to perform administrative and development tasks on SQL Server 2005. SSMS is a replacement for the Enterprise Manager, Query Analyzer, and Analysis Manager that were available in SQL Server 2000. Microsoft has consolidated all those tools into one, with a focus on providing a tool that suits the needs of both developers and database administrators (DBAs).

SSMS is a complicated tool that provides an entry point to almost all of SQL Server's functionality. The functionality that is accessible from SSMS is entirely too much to cover in one chapter. The aim of this chapter is to give a basic overview of SSMS, with a concentration on features that are new to SQL Server 2005. Others chapters in this book focus on the components of SSMS and provide more detailed coverage.

What's New in SSMS

SSMS is an entirely new environment for SQL Server 2005. It encapsulates many of the features previously available in other tools and also offers many new features. The bulk of these new features can be grouped into four major categories: environmental changes, integrated management, enhanced query authoring, and enhanced project management.

The environmental changes are changes that have occurred to the graphical application. SSMS has a new look and feel, and it offers some significant change to the way that

windows are managed within the application. The application was rewritten in .NET and has features that are more like the development environment found in Visual Studio. Many windows in SSMS are dockable, can be pinned, and can be set to Auto Hide. In addition, many of the management dialog boxes are now modal, which means they can stay open while you open other windows within the application.

The new integrated management features stem from a consolidation of management tools. SSMS now contains management functionality that was contained in SQL Server 2000's Enterprise Manager, Analysis Manager, SQL Server Service Manager, Query Analyzer, and other tools. The functionality from these tools has been integrated into one environment that shares common Help, a summary window that displays useful information, an Object Explorer tree for easy navigation, and a myriad of other tools that can be accessed from one central location.

The changes related to query authoring are also based on a consolidation of functionality that was contained in several different tools in previous versions of SQL Server. Scripts that were previously created with Query Analyzer or Analysis Services can now be authored in SSMS. A new SSMS window named the Query Editor is an editing tool for the creation of SQL Server scripts. It brings with it many of the great features from the prior tools, such as color coding, syntax checks, and performance analysis, along with some new features, such as Dynamic Help, an XML editor, enhanced templates, and the ability to write scripts without being connected to the database.

The last category of changes in SSMS relates to managing the files or scripts you create when working with SQL Server 2005. SSMS provides a tool to organize scripts, connections, and other, related files into projects. These projects can also be grouped to form a solution. Once again, this functionality is based on the Visual Studio application development environment and the way it organizes development files into projects and solutions. As with Visual Studio, these files can also be managed with source control in SSMS. SSMS provides links to Visual SourceSafe, which allows you to secure the files and manage version control.

This chapter further explores the new features in SSMS. It first examines the features at the environmental level, focusing on how SSMS behaves and how to best utilize the environment. Next, it looks at the administrative tools and what changes have been made to help you better manage your SQL Server environment. Finally, this chapter looks at the development tools that are available with SSMS and the changes that have been made to improve your SQL Server development experience.

The Integrated Environment

Those who have been working with SQL Server for a long time may remember the SQL Enterprise Manager that came with SQL Server 6.5. In some respects, with SSMS, Microsoft has moved back to the paradigm that existed then. Like the SQL Server 6.5 Enterprise Manager, SSMS provides an integrated environment where developers and DBAs alike can perform the database tasks they need. Say goodbye to Query Analyzer, Analysis Manager, and a number of other desperate tools and say hello to SSMS, which provides "one-stop shopping" for most of your database needs.

Window Management

Figure 3.1 shows a sample configuration for the SSMS main display. The environment and the windows that are displayed are completely customizable, with the exception of the document window area. Figure 3.1 shows the document window area displaying the Summary page. The Summary page is the default, but other pages, such as a Query Editor window, can take the focus in this tab-oriented section of the SSMS display.

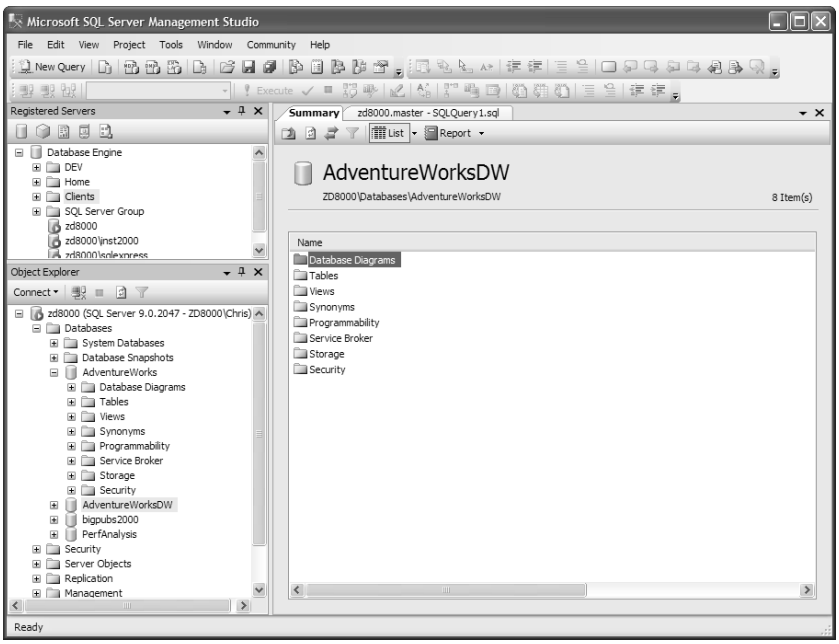

FIGURE 3.1 The SSMS main display.

The dialogs that form the rest of the SSMS display are referred to as *components* and include the Registered Servers and Object Explorer windows that are shown in Figure 3.1, as well as a number of other components that can be displayed via the View menu found at the top of the SSMS display. You can configure each of the component windows in a number of ways; for example, you can have them float, or you can hide, dock, Auto Hide, or display them as tabbed documents in the document window area.

The configuration that you choose for your SSMS display depends on the type of work you do with SQL Server as well as the type of person you are. The Auto Hide feature causes the component window to shrink to a tab along the left or right side of the display. When you mouse over the tab, the window automatically expands and stays expanded as long as the mouse cursor remains in the component window area. Auto Hide helps maximize the working real estate that is available in the document window for query development and the like. Docking many windows can clutter the screen, but it allows you to view many different types of information all at once. This is a matter of personal preference, and SSMS has made it very easy to change.

TIP

You can reposition the component windows by dragging and dropping them to the desired locations. When you are in the middle of a drag and drop, rectangular icons with arrows are displayed at different locations on the SSMS window surface. If you mouse over one of these arrowed icons to select the window location, you see the window destination highlighted. If you release your mouse button while the destination is highlighted, the window docks in that position.

Some users at first ignore the arrow icons and keep hovering the window over the location where you want the window to go. Hovering the window over the desired location does not allow you to effectively dock it. You should save yourself some time and aggravation and use the arrow icons for drag-and-drop positioning.

The other big changes to the SSMS window environment include non-modal windows that are sizable. The change to non-modal windows allows you to perform multiple tasks at once without needing to open another instance of the SSMS application. Enterprise Manager users of SQL Server 2000 were forced to open another instance of the application during many administrative tasks in order to continue with other work. With SSMS, you can launch a backup with the Back Up Database dialog box and then continue working with the Object Explorer or other components in SSMS while the backup is running. This is a great timesaver and helps improve overall productivity.

The ability to size the dialog boxes is another user-friendly change that may seem minor but is quite handy on certain windows. For example, the SQL Server 2000 Enterprise Manager Restore dialog had a fixed size. Viewing the backup set information in this relatively small (nonsizable) dialog box was a challenge. The Restore dialog in SQL Server 2005's SSMS can contain a slew of information related to the backup sets available for restore. The ability to size the windows allows for much more information to be displayed.

The tabbed document window area provides some usability improvements as well. This area, as described earlier, is fixed and is always displayed in SSMS. Component windows can be displayed in this area, along with windows for the Query Editor, diagrams, and other design windows. If desired, you can change the environment from a tabbed display to multiple-document interface (MDI) mode. In this mode, each document is opened in its own window within the document window. The MDI mode manages windows like the SQL Server 2000 Query Analyzer and may be more user-friendly for some people. You can change to MDI mode by selecting Tools, Options and then selecting MDI Environment from the General page.

One particularly useful window that can be displayed in the document window is the Summary page. This new window displays information relative to the node that is selected in the Object Explorer and includes options to produce detailed reports and graphs. The Summary page is displayed in the document window by default when SSMS is launched, but you can also display it by pressing F7 or choosing Summary from the View menu.

TIP

In SQL Server 2000, you could select multiple objects for scripting by selecting the items from the Object Explorer tree in Enterprise Manager. You cannot use the Object Explorer tree to do this with SQL Server 2005, and this has generated some confusion. The solution is the Summary page, which provides a means for doing multiple selections of the objects it displays. You can hold down the Ctrl key and click only those items that you want to script. After you have selected the items you want, you simply right-click one of the selected items and choose the preferred scripting method. This also works with scheduled jobs that are displayed in the Summary page. SQL Server 2000 did not offer this capability.

The reports that are available on the Summary page are often overlooked. Part of the reason for this may be that the reports are not available for every node in the Object Explorer tree. Top-level nodes in the tree are where most of the reports are found. For example, if you select a database in the Object Explorer tree and view the Summary page, you see a Report icon that is enabled on the toolbar at the top of the Summary page. If you click the drop-down arrow next to that icon, you find a list of reports that are available for creation. These reports include Disk Usage, Backup and Restore Events, Top Transactions by Age, and a host of others. Graphs are included with some reports, and you can export or print all these reports. Figure 3.2 shows an example of the Disk Usage report for the `AdventureWorks` database.

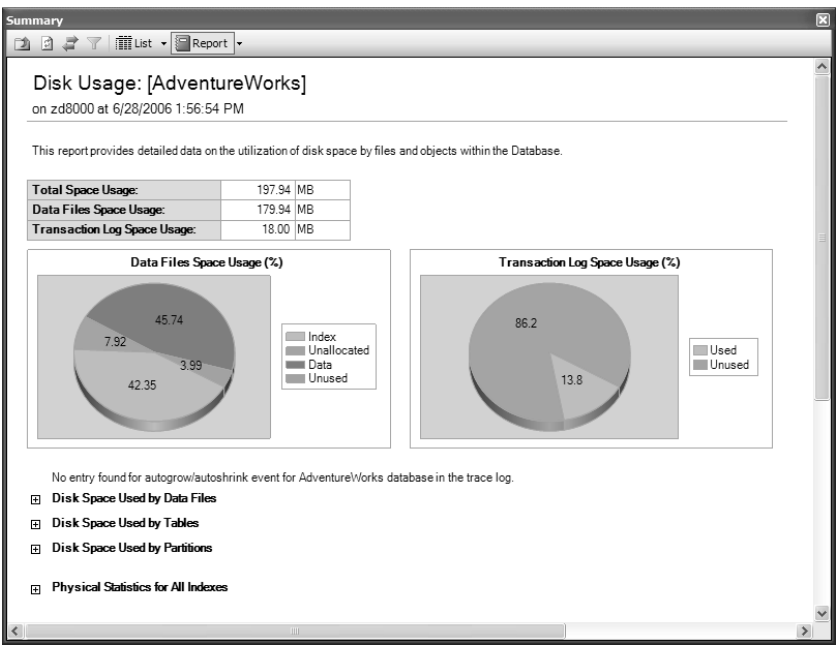

FIGURE 3.2 A Disk Usage summary report.

The graphs are easy to read, and some sections of the report can be expanded to provide more detail. Bullets at the bottom of a report are nodes that can be expanded. For example, the bullets Disk Space Used by Data Files and Disk Space Used by Table at the bottom of Figure 3.2 can be expanded.

Integrated Help

SSMS offers an expanded set of help facilities as well as improved integration into the application environment. The Help sources have been expanded to include both local and online resources. Local help is similar to the Help resources available in past versions and references files that are installed on your machine during the installation process. Local help includes the local SQL Server Books Online resources. Local help files are static and get updated only if another documentation installation is run on the local machine.

Online help is new to SQL Server 2005 and provides access to content that is not static and can be updated with the very latest changes. Three default online resources are provided by default:

- **MSDN Online**—MSDN Online contains the latest version of the MSDN documentation, including the latest quarterly releases.
- **Codezone Community**—Codezone Community includes a set of third-party websites that have partnered with Microsoft and provide a wealth of information from sources outside Microsoft.
- **Questions**—The Questions option allows you to search the forum archives for answers to questions that others have already asked. It also allows you to post your own questions.

The help resources you use on your machine are configurable. You can choose to search online resources first, followed by local help, or you can choose an option that searches local help resources first, followed by online resources. You can also choose specific Codezone online resources to search, or you can eliminate the search of all online resources. Figure 3.3 shows the online help Options window, which allows you to configure your Help options. You access this dialog by selecting Tools, Options.

The Help resources you select are used when you search for content within the Help facility. When you use both local and online resources options, you see results from multiple locations in your search results. Figure 3.4 shows a sample Books Online Document Explorer window with results from a search on “Management Studio.” Notice that the panel on the right side of the window lists entries under Local Help, MSDN Online, Codezone Community, and Questions. Each of these sections contains search results that you can access by simply clicking on that area. The number of search results for each section is displayed in parentheses after the section name.

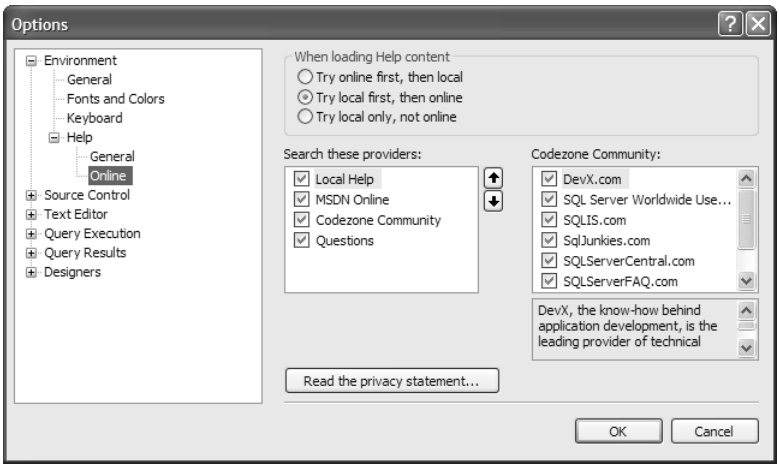

FIGURE 3.3 Setting Help options.

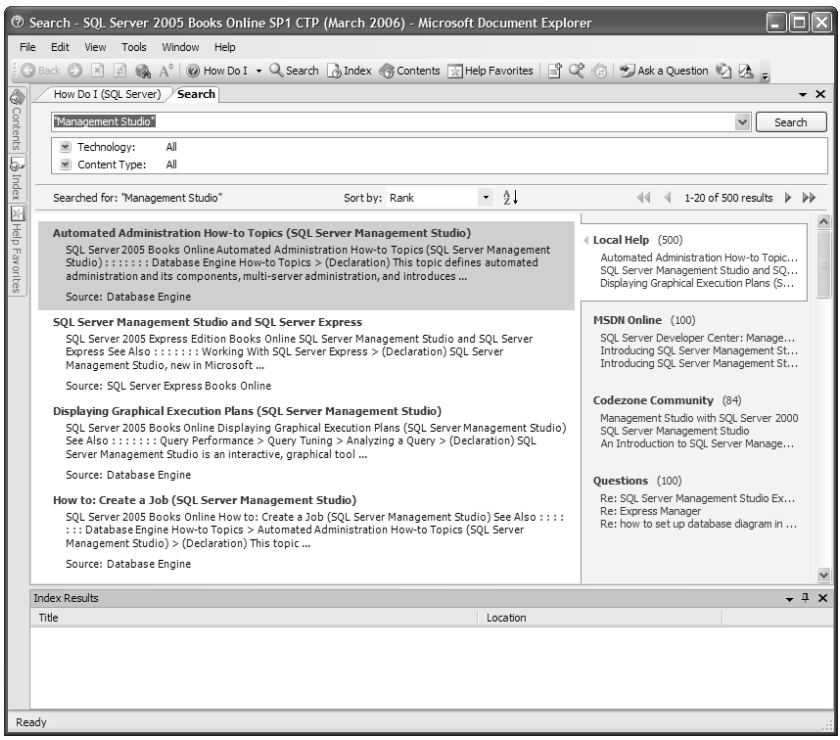

FIGURE 3.4 A Books Online search.

One other significant change to the help facilities in SSMS is the addition of Dynamic Help. Dynamic Help is a carryover from the Visual Studio environment. It is a help facility that automatically displays topics in a Help window that are related to what you are doing in SSMS. For example, if you are working in a query window and type the word `SELECT` to start your query, the Dynamic Help window displays several topics related to the `SELECT` statement. If you are working in the Object Explorer, it displays Help topics related to the Object Explorer.

Dynamic Help is one of the component windows that you can dock or position on the SSMS surface. To use Dynamic Help, you select Help, Dynamic Help. Figure 3.5 shows an example of the SSMS environment with the Dynamic Help window docked on the right side of the window. The Dynamic Help topics in this example are relative to the `SELECT` keyword that is typed in the Query Editor window in the middle of the screen.

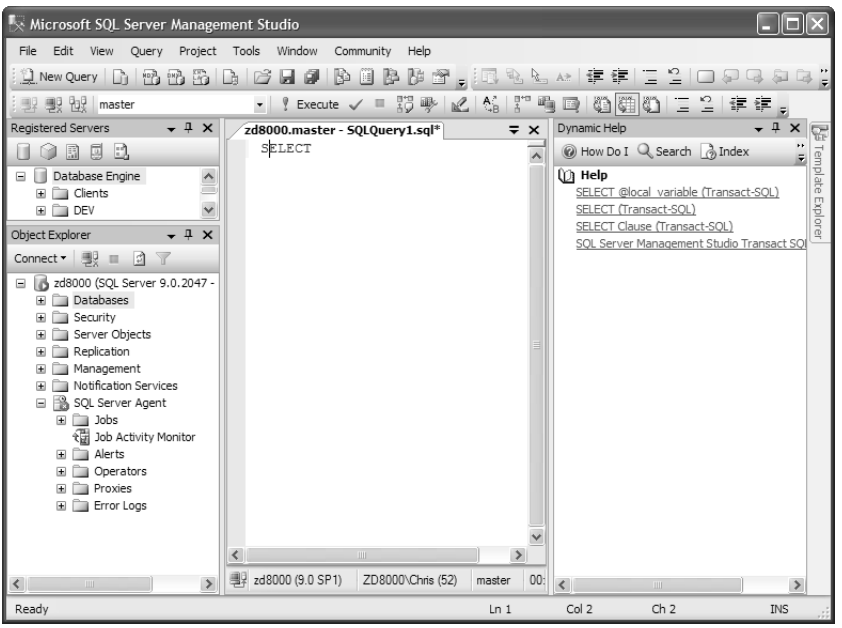

FIGURE 3.5 Dynamic Help.

Administration Tools

The tools that are available with SSMS can be broadly categorized into tools that are used for administering SQL Server and tools that are used for developing or authoring new SQL Server objects. As a matter of practice, developers use some of the administrative tools, and administrators use some of the development tools.

SSMS comes with an expanded set of tools to help with SQL Server administrative tasks. It builds on the functionality that was available in the SQL Server 2000 Enterprise Manager and adds some new tools and functionality to help ease the administrative burden.

Using Registered Servers

Registered servers is a new concept in SQL Server 2005 and represents a new division between managing servers and registering servers. With the SQL Server 2000 Enterprise Manager, the Microsoft Management Console (MMC) tree was displayed on the left side of the Enterprise Manager screen, and it contained servers that had been registered via that tree. Any registered servers or groups were listed in the tree, along with any of the associated objects.

With SQL Server 2005, registered servers are managed and displayed in the Registered Servers component window. The objects associated with these registered servers are displayed in the Object Explorer rather than in the Registered Servers window.

Figure 3.6 shows an example of the Registered Servers window, with several server groups and their associated registered servers. You can add new groups any time; this window offers a handy way of organizing the servers you work with.

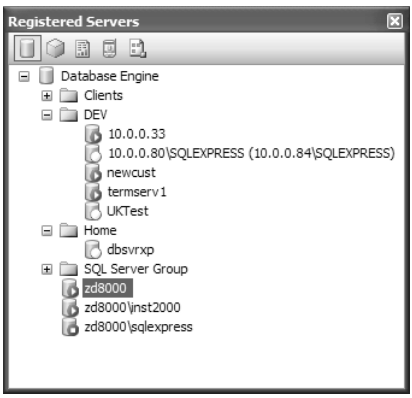

FIGURE 3.6 The Registered Servers window.

The servers listed in Figure 3.6 are all database engine servers. These server types are the conventional SQL Server instances, like those you could register in the SQL Server 2000 Enterprise Manager. You can also register several other types of servers. The icons across the top of the Registered Servers window indicate the types of servers that can be registered. In addition to database engine servers, you can also register servers for Analysis Services, Reporting Services, SQL Server Mobile, and Integration Services. The Registered Servers window gives you one consolidated location to register all the different types of servers that are available in SQL Server 2005. You simply click the icon associated with the appropriate server type, and the registered servers of that type are displayed in the Registered Servers tree.

NOTE

The SQL Server 2005 Registered Servers window enables you to register servers that are running SQL Server 2000 and SQL Server 7.0 as well. You can manage all the features of SQL Server 2000 with SQL Server 2005 tools. You can also have both sets of tools on one machine. The SQL Server 2000 and SQL Server 2005 tools are compatible and function normally together.

The SQL Server 2000 Enterprise Manager and Query Analyzer cannot be used to manage SQL Server 2005. You can connect the Query Analyzer to a SQL Server 2005 instance and run queries, but the Object Explorer and other tools are not compatible with SQL Server 2005.

When a server is registered, you have several options available for managing the server. You can right-click the server in the Registered Servers window to start or stop the related server, open a new Object Explorer window for the server, connect to a new query window, or export the registered servers to an XML file so that they can be imported on another machine.

TIP

The import/export feature can be a real timesaver, especially in environments where many SQL servers are managed. You can export all the servers and groups that are registered on one machine and save the time of registering them all on another machine. For example, you can right-click the `Database Engine` node, select Export, and then choose a location to store the XML output file. Then, all you need to do to register all the servers and groups on another machine is move the file to that machine and import the file.

Using Object Explorer

The Object Explorer window that existed in the SQL Server 2000 Query Analyzer has been integrated into SSMS. It has the same tree-like structure that was present in SQL Server 2000 but contains some significant improvements over its predecessor. The most significant feature for those folks managing a large number of database objects is the ability to populate the Object Explorer tree asynchronously. This may not hit home for folks who deal with smaller databases, but those who waited on the synchronous population of Object Explorer in SQL Server 2000 will be excited. The Object Explorer tree in SSMS displays immediately and allows navigation in the tree and elsewhere in SSMS while the population of the tree is taking place.

The Object Explorer is adaptive to the type of server that it is connected to. For a database engine server, the databases and objects such as tables, stored procedures, and so on are displayed in the tree. If you connect to an Integration Services server, the tree displays information about the packages that have been defined on that type of server. Figure 3.7 shows an example of the Object Explorer with several different types of SQL Server servers displayed in the tree. Each server node has a unique icon that precedes the server name, and the type of server is also displayed in parentheses following the server name.

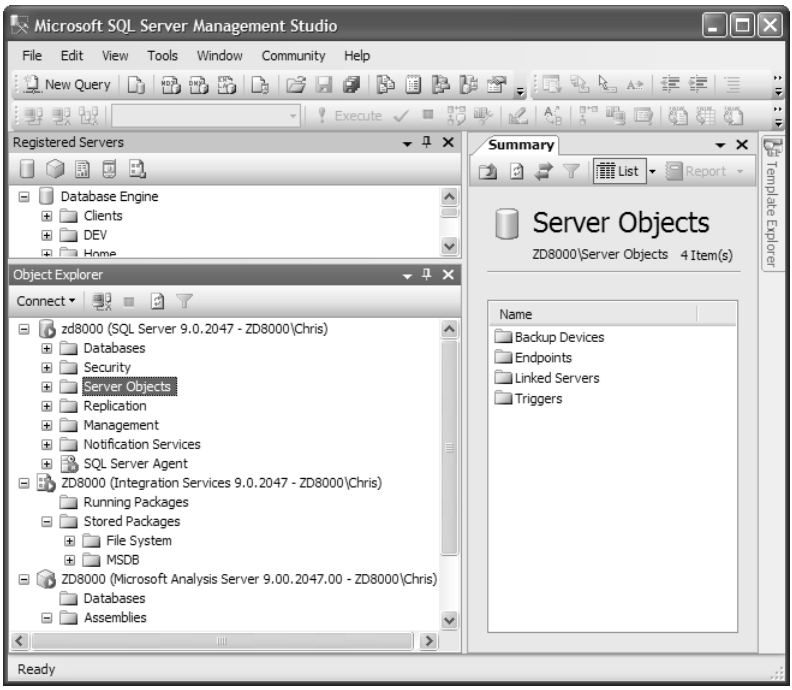

FIGURE 3.7 Multiple server types in Object Explorer.

The objects displayed in the Object Explorer tree can be filtered in SQL Server 2005. The number of filters is limited, but those that are available can be helpful. For example, you can filter the tables that are displayed in Object Explorer based on the name of the table, the schema that it belongs to, or the date on which it was created. Again, for those who deal with large databases and thousands of database objects, this feature is very helpful.

Administrators also find the enhanced scripting capabilities in the Object Explorer very useful. The scripting enhancements are centered mostly on the administrative dialog boxes. These dialogs now include a script button that allows you to see what SSMS is doing behind the scenes to effect your changes. In the past, the Profiler could be used to gather this information, but it was more time-consuming and less integrated than what is available now.

Figure 3.8 shows an example of an administrative dialog, with the scripting options selected at the top. You can script the commands to a new query window, a file, the Windows Clipboard, or a job that can be schedule to run at a later time.

Aside from these features, many of the features and much of the functionality associated with the Object Explorer is similar to what was found in SQL Server 2000. Keep in mind that there are some additional nodes in the Object Explorer tree and that some of the objects are located in different places. There is now a separate node for the SQL Server Agent that contains scheduled jobs and related objects. Linked servers are now located under the `Server Objects` node, and several new additions are available in the `Management` node that were found elsewhere in the prior version.

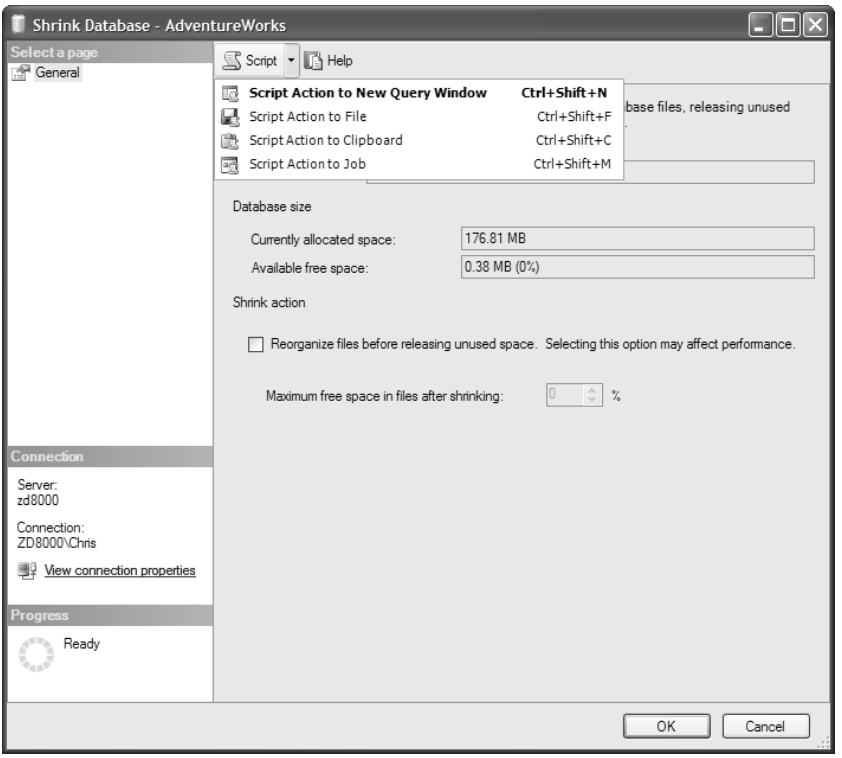

FIGURE 3.8 Scripting from administrative dialogs.

Using Activity Monitor

The functionality that was previously available in the `Current Activity` node of Enterprise Manager has been ported to a new tool named the Activity Monitor. The Activity Monitor is a non-modal window that is launched from the Object Explorer tree. To access it, you open the `Management` node of the tree and double-click the `Activity Monitor` node. Figure 3.9 shows an example of the Activity Monitor window.

The default Process Info page lists current processes on the database server. By default, the system processes are not listed, but you can add them to the display by using the filtering capabilities in the application. If you click the Filter button, a Filter Settings window like the one shown in Figure 3.10 is displayed. You can set the `Show System Processes` value to `True` to display all the processes, and you can adjust any of the other filter values to display the desired set of processes.

The other two pages in the Activity Monitor display information about locks on the server. You can display locks by server process ID (SPID) or select locks based on a specific database object. This information is similar to what is retrieved when you run the `sp_lock` system stored procedure or the `sys.dm_tran_locks` dynamic management view.

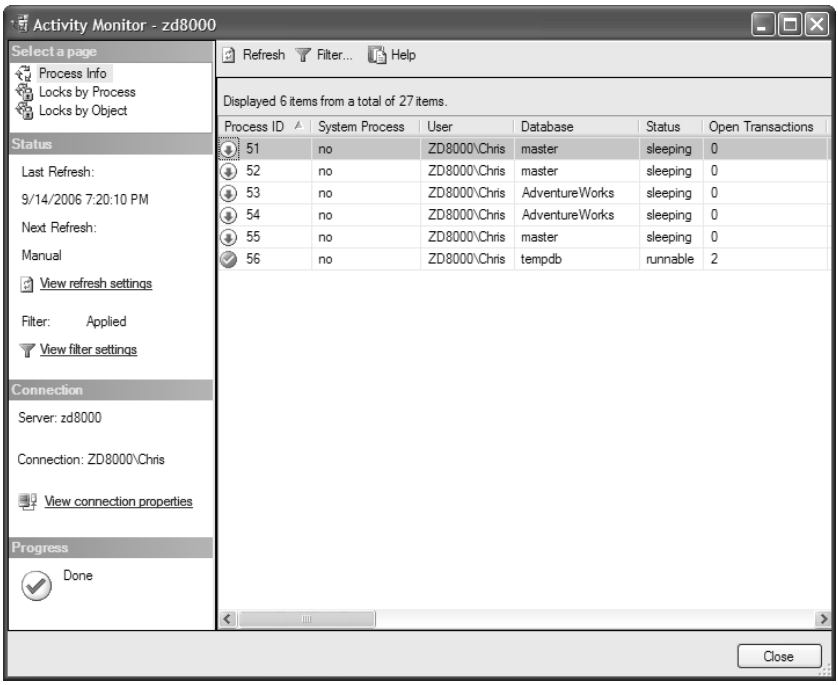

FIGURE 3.9 Process info in the Activity Monitor.

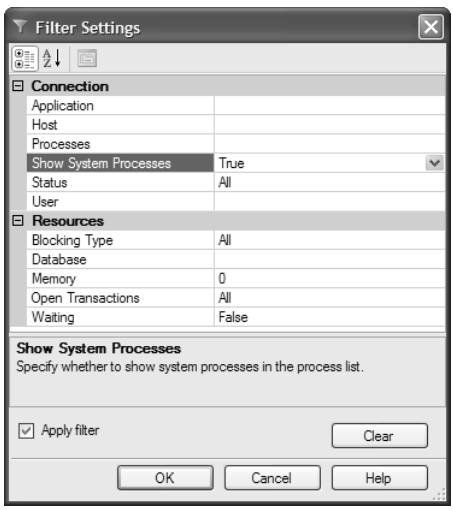

FIGURE 3.10 Filter settings in the Activity Monitor.

NOTE

Each of the pages in the Activity Monitor has an abundance of columns that display useful information. The default window is not nearly big enough to display all the information. You can use the scrollbar to navigate to the columns that are hidden. To rearrange the order of the columns, you simply drag a column header to the desired location.

One of the most impressive features of the Activity Monitor is its ability to refresh the display automatically. You can click the View Refresh settings option on the left side of the screen to adjust the refresh rate. You can select the Auto Refresh Every option and select the number of seconds between refreshes to have the screen automatically refresh.

Using Log File Viewer

The Log File Viewer is another non-modal window that is new to SQL Server 2005. Like the Activity Monitor, it houses information that was previously displayed in the document window in the SQL Server 2000 Enterprise Manager. It can display log files that are generated from several different sources, including Database Mail, SQL Server Agent, SQL Server, and Windows NT.

The Log File Viewer can be launched from the related node in the SSMS Object Explorer. For example, you can select the `Management` node and expand `SQL Server Error Logs`. If you double-click one of the error logs listed, a new Log File Viewer window is launched, displaying the SQL Server log file entries for the log type selected (see Figure 3.11).

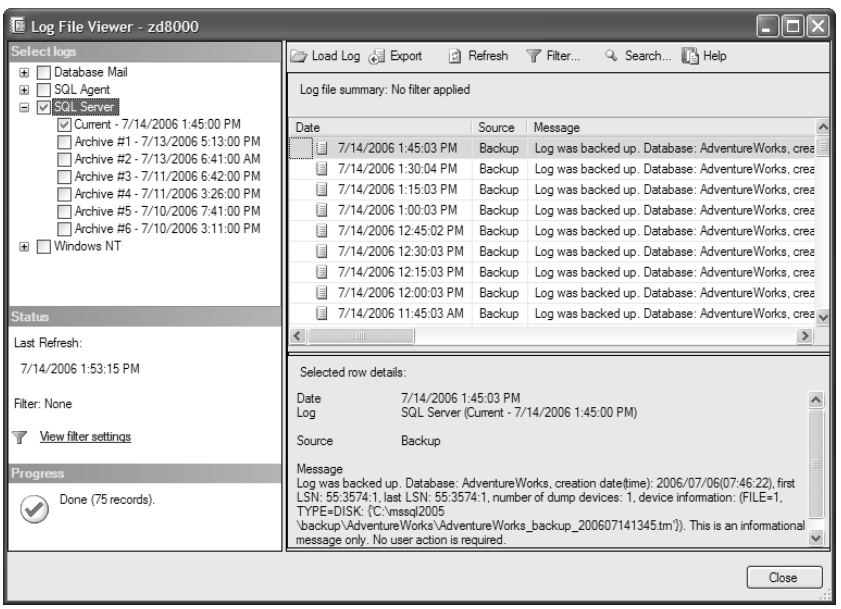

FIGURE 3.11 SQL Server logs displayed in the Log File Viewer.

NOTE

By default, entries are shown in the SQL Server Log File Viewer from newest to oldest. This is different from the default order in the SQL Server 2000 Enterprise Manager, which displayed the log file entries from oldest to newest.

One of the first things you notice when you launch the Log File Viewer is that there is a tree structure at the top-left corner of the screen that shows the log files you are viewing. You can see that there are four different log types available: Database Mail, SQL Agent, SQL Server, and Windows NT. You can choose to display multiple log files within a given log type (for example, the current SQL Server log and Archive #1) or you can select logs from different sources. For example, you can display all the current log entries for SQL Server and the current log entry for the SQL Server Agent.

When multiple logs are selected, you can differentiate between the rows shown on the right side of the Log File Viewer by looking at the Log Source column and the Log Type column. The Log Source values match up with the names that are shown in the tree structure where the log was selected. The Log Type column shows the type of log, such as SQL Agent or SQL Server. Rows from the different log types are displayed together and sorted according to the date on which the row was created. The sort order cannot be changed.

TIP

You can rearrange the order of the columns shown in the Log File Viewer. You simply click the column header and drag the column to the desired location. When viewing rows for more than one log type or multiple logs, it is best to drag the Log Type and Log Source columns to a location that is easily viewed so that you can distinguish between the entries.

Other noteworthy features in the Log File Viewer include the ability to filter and load a log from an external source. You can filter on dates, users, computers, the message text, and the source of the message. You can import log files from other machines into the view by using the Load Log facility. This works hand-in-hand with the Export option, which allows you to export the log to a file. These files can be easily shared so that others can review the files in their own Log File Viewer.

Development Tools

SSMS delivers an equally impressive number of enhancements for database developers. These new tools are based on tools such as Query Analyzer that were available in prior versions of SQL Server. They deliver the same functional value available in prior releases and offer enhancements that address some of the shortcomings.

The Query Editor

The Query Editor sits at the top of the list for new development tools in SSMS. The Query Editor, as its name indicates, is the editing tool for writing queries in SSMS. It contains

much of the functionality that was contained in SQL Server 2000's Query Analyzer. The ability to write Transact-SQL (T-SQL) queries, execute them, return results, generate execution plans, and many of the other features you may be familiar with in Query Analyzer are also available with the Query Editor.

One main difference with the Query Editor is that is has been integrated into the SSMS environment. In SQL Server 2000, the Query Analyzer was a separate application with its own independent interface. In SQL Server 2005, SSMS houses the query-editing capabilities along with all the administrative capabilities.

NOTE

The biggest upside to the integration of the query-editing tool into the SSMS environment is that you can find almost anything you need to administer or develop on your SQL Server database in one spot. There is no need to jump back and forth between applications. One possible downside, however, is that SSMS may be much more than some database developers need.

Clicking the New Query button, opening a file, and selecting the Script to File option from a list of database objects in the Object Explorer are just a few of the ways to launch the Query Editor. Figure 3.12 shows the Query Editor window with a sample `SELECT` statement from the `AdventureWorks` database. The figure shows the Query Editor window displayed on the right side of the screen and the Object Explorer on the left side.

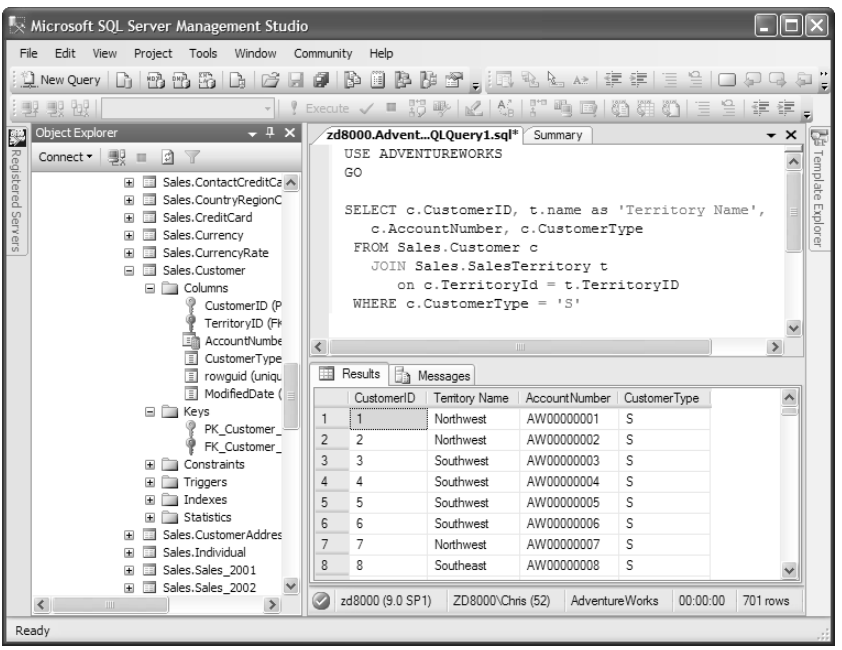

FIGURE 3.12 The Query Editor window in SSMS.

The basic editing environment within the Query Editor is similar to Query Analyzer. The top portion of the Query Editor window contains the query. The bottom portion contains the results of an executed query. The results can be displayed as text, displayed in a grid format, or output as XML. However, in the Query Editor, windows are by default managed differently than with Query Analyzer. Multiple Query Editor windows are displayed in a tabbed format; in comparison, Query Analyzer displayed a separate window for each query.

TIP

The tabbed document display has some advantages, but you can set an option in SSMS that causes the Query Editor to behave much like the Query Analyzer. To do this, you select Tools, Options to launch the Options dialog. The default page has a section named Environmental Layout. If you choose the MDI Environment option, you set SSMS in MDI mode instead of the tabbed layout.

Query Editor Types

The Query Editor in SQL Server 2005 enables you to develop different types of queries. You are no longer limited to database queries based on SQL. You can use the Query Editor to develop all types of SQL Server Scripts, including those for SQL Server Analysis Services (SSAS) and SQL Server Mobile Edition. The SSAS queries come in three different flavors: multidimensional expressions (MDX), data mining expressions (DMX), and XML for analysis (XMLA). Only one selection exists for creating SQL Server Mobile Edition scripts.

You see these new query options when you create a new query. When you select New from the SSMS menu, you can choose what type of query to create. You use the Database Engine Query choice to create a T-SQL query against the database engine. The other new query options correspond to SSAS and SQL Server Mobile Edition. The toolbar on SSMS also has icons that correspond to each type of query that can be created.

Each query type has a code pane that works much the same way across all the different types of queries. The code pane, which is the topmost window, color-codes the syntax that is entered, and it has sophisticated search capabilities and other advanced editing features that make it easy to use. The features that are new to SQL Server 2005 and apply to all the editor types include line numbering, bookmarks, hyperlinks in the comments, and a color-coded indicator that is shown in front of each line that has changed since the script was opened.

Other code pane features are available only for certain types of queries. IntelliSense, which automatically completes syntax and arguments, is available for all queries except database engine queries. Squiggles, which are wavy lines that appear below a word in the editor to indicate possible syntax errors, are available with MDX, DMX, and XML queries. The MDX, DMX, and XML editors also offer code outlining, which enables you to expand and collapse code segments to make it easier to review code.

Disconnected Editing

New to SQL Server 2005 is the ability to use the code editor without a database connection. When creating a new query, you can choose to connect to a database or select Cancel to leave the code pane disconnected. To connect to the database at a later time, you can right-click in the code pane window and select the Connect option. You can also disconnect the Query Editor at any time or choose the Change Connection option to disconnect and connect to another database all at once.

Along with disconnected editing are some changes to the Windows behavior that are worth noting. The biggest changes relate to the behavior of query windows that are currently open at the time that a file is opened for editing. With SQL Server 2000 Query Analyzer, the currently selected window would be populated with the contents of the file that you were opening. Prior to this replacement, a prompt would be displayed that asked whether you wanted to save your results. If the query window was empty, the contents would be replaced without the prompt for saving.

With SQL Server 2005, a new query window is opened every time a new file is opened. The new window approach is faster but can lead to many more open windows in the document window. You need to be careful about the number of windows/connections you have open. Also, you need to be aware that the tabbed display shows only a limited number of windows. Additional connections can exist even if their tabs are not in the active portion of the document window.

Editing SQLCMD Scripts in SSMS

SQLCMD is a command-line utility that is new to SQL Server 2005. You can use it for ad hoc interactive execution of T-SQL statements and scripts. It is basically a replacement for the `ISQL` and `OSQL` commands that were used in prior versions of SQL Server. (`OSQL` still works with SQL Server 2005, but `ISQL` has been discontinued.)

What's new to SSMS is the ability to write, edit, and execute SQLCMD scripts within the Query Editor environment. The Query Editor in SSMS treats SQLCMD scripts in much the same way as other scripts. The script is color-coded and can be parsed or executed. This is possible only if you place the Query Editor in SQLCMD mode, which you do by selecting Query, SQLCMD Mode or selecting the SQLCMD mode icon from the SSMS toolbar.

Figure 3.13 shows a sample SQLCMD script in SSMS that can be used to back up a database. This example illustrates the power and diversity of a SQLCMD script that utilizes both T-SQL and SQLCMD statements. It uses environment variables that are set within the script. The script variables `DBNAME` and `BACKUPPATH` are defined at the top of the script with the `SETVAR` command. The `BACKUP` statement at the bottom of the script references these variables, using the convention `$(variablename)`, which substitutes the value in the command.

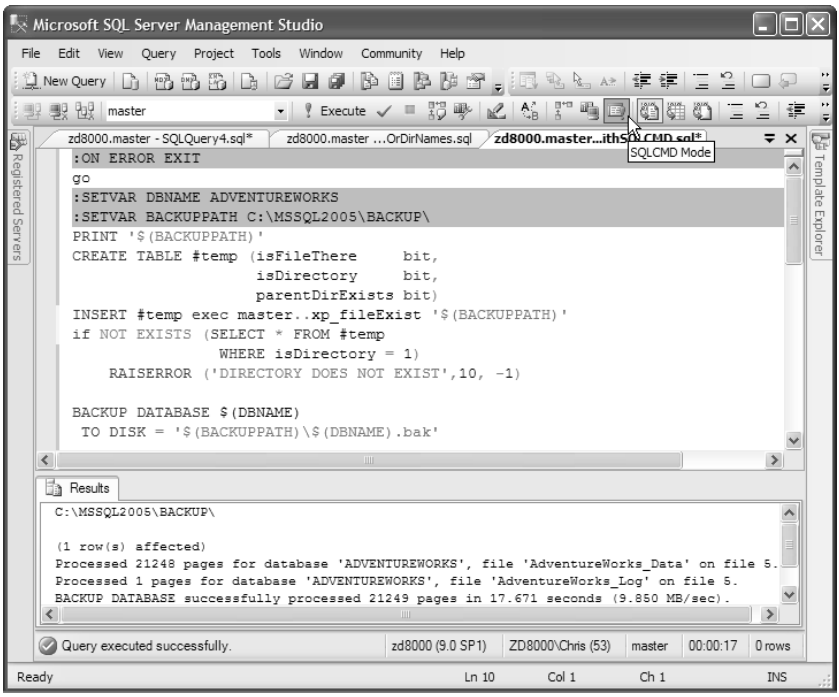

FIGURE 3.13 Editing a SQLCMD script in SSMS.

SQLCMD scripts that are edited in SSMS can also be executed within SSMS. The results are displayed in the results window of the Query Editor window, just like any other script. After you test a script, you can execute it by using the SQLCMD command-line utility. The SQLCMD command-line utility is a very powerful tool that can help automate script execution. For more information on using SQLCMD in SSMS, refer to the Books Online topic "Editing SQLCMD Scripts with Query Editor." The SQLCMD command-line utility is discussed in more detail in Chapter 4, "SQL Server Command-Line Utilities."

Regular Expressions and Wildcards in SSMS

SSMS has a robust search facility that includes the use of regular expressions. Regular expressions provide a flexible notation for finding and replacing text, based on patterns within the text. Regular expressions are found in other programming languages and applications, including the Microsoft .NET Framework. The regular expressions in SSMS work in much the same way as these other languages, but there are some differences in the notation.

The option to use regular expressions is available whenever you are doing a find or replace within an SSMS script. You can use the find and replace option in the code pane or the results window. You can use the Find and Replace option from the Edit menu or use press either the Ctrl+F or Ctrl+H shortcut keys to launch the Find and Replace dialog

box. Figure 3.14 shows an example of the Find and Replace dialog that utilizes a regular expression. This example is searching for the text Customer, preceded by the @ character and not followed by the Id characters. This kind of search could be useful for searching a large stored procedure where you want to find the customer references but don't want to see the variables that contain *customer* in the first part of the variable name.

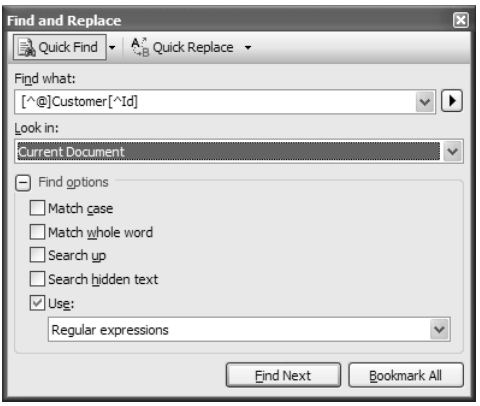

FIGURE 3.14 A find and replace with regular expressions.

You use regular expressions only when the Use check box in the Find and Replace dialog is selected. When this option is selected, you can choose either Regular Expressions or Wildcards. Wildcard searches work much the same way in SSMS as they do in file searches. For example, if you wanted to find any references to the word *zip*, you could enter `*zip*` in the Find What text box. The wildcard options are limited but very effective for simple searches.

Regular expressions have a much more extensive number of available search options. When you choose the option to use regular expressions, the arrow button is enabled to the right of the text box where you enter your search text. If you click this button, you are given an abbreviated list of regular expression characters that you can use in your searches. A brief description of what each character represents in the search is listed next to the character. For a complete list of characters, you can choose the Complete Character List option at the bottom of the list. This option brings you to the Books Online topic "How to: Search with Regular Expressions," which gives a comprehensive review of all the characters.

Enhanced Performance Output

The Query Editor in SSMS has an extended set of options available for capturing and distributing performance-related data. It contains many of the familiar performance features that you may have grown accustomed to in SQL Server 2000 Query Analyzer—plus more.

Changes in the collection of performance data include a new Execution Plan tab that is displayed in the results window, along with the Results and Messages tab. The Execution Plan tab can be populated with two different types of plans: estimated plans and actual plans. The actual execution plan is a new display for SQL Server 2005; it shows the plan that was used in generating the actual query results. The actual plan is generated along with the results when the Include Actual Execution Plan option is selected. This option can be selected from the SSMS toolbar or from the Query menu. Figure 3.15 shows an example of an actual execution plan generated for a query against the `AdventureWorks` database. It uses the familiar treelike structure that was also present in SQL Server 2000, but the display has been enhanced for SQL Server 2005. The ToolTips that are displayed when you mouse over a node in the execution plan include additional information; you can see that information in a more static form in the Properties window if you right-click the node and select Properties. The icons in the graphical plan have changed, and the display is generally easier to read in SQL Server 2005.

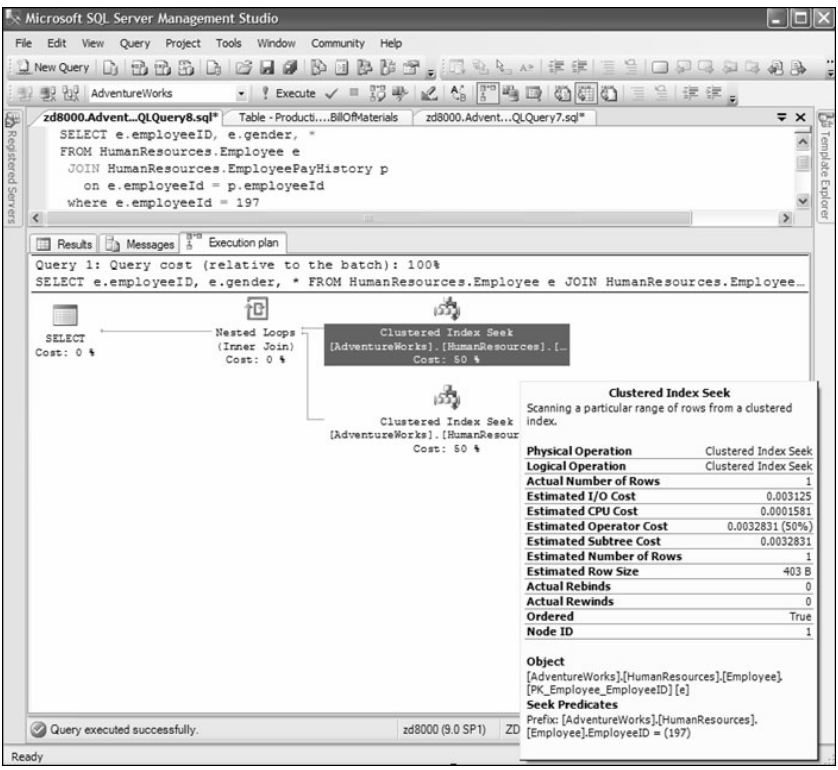

FIGURE 3.15 Displaying an actual execution plan in Query Editor.

NOTE

The Manage Indexes and Manage Statistics options that were available in the SQL Server 2000 Query Analyzer are not present in the Query Editor in SQL Server 2005. Those options in Query Analyzer were accessible by right-clicking a node in the query plan. You can use the Database Engine Tuning Advisor (DTA) in SQL Server 2005 to analyze the Query Editor statements or open the Table Designer to manage the indexes on a specific table.

Query plans generated in the Query Editor are easier to distribute in SQL Server 2005. You have several options for capturing query plan output so that you can save it or send it to someone else for analysis. If you right-click an empty section of the Execution Plan window, you can select the Save Execution Plan As option, which allows you to save the execution plan to a file. By default, the file has the extension `.sqlplan`. This file can be opened using SSMS on another machine to display the graphical output.

The query plan can also be output in XML format and distributed in this form. You make this happen by using the `SET SHOWPLAN_XML ON` option. This option generates the estimated execution plan in a well-defined XML document. The best way to do this is to turn off the display of the actual execution plan and execute the `SET SHOWPLAN_XML ON` statement in the code pane window. Next, you set the Query Editor to return results in grid format and then execute the statements for which you want to generate a query plan. If you double-click the grid results, they are displayed in the SSMS XML editor. You can also save the results to a file. If you save the file with the `.sqlplan` extension, the file will display the graphical plan when opened in SSMS.

Using the Query Designer in the Query Editor

A graphical query design tool is now accessible from the Query Editor window where you write your queries. This is a great option that was missing in prior versions of SQL Server. With SQL Server 2000, you could access a graphical query designer by opening a table in Enterprise Manager and selecting Query, but this option was disconnected from the Query Analyzer environment, where the queries were authored.

With SQL Server 2005, you can right-click in the Query Editor window and choose Design Query in Editor. A dialog box appears, allowing you to add tables to the graphical query designer surface. The tables that are selected are shown in a window that allows you to select the columns you want to retrieve. Columns that are selected appear in a `SELECT` statement that is displayed at the bottom of the Query Designer window. Figure 3.16 shows an example of the Query Designer window that contains two tables from the `AdventureWorks` database. The two tables selected in this figure are related, as indicated by the line between them.

The T-SQL statements are generated automatically as you select various options on the Query Designer screen. If you select Sort Type, an `ORDER BY` clause is added. If you choose an alias for a column, it is reflected in the T-SQL. If tables are related, the appropriate joins are generated.

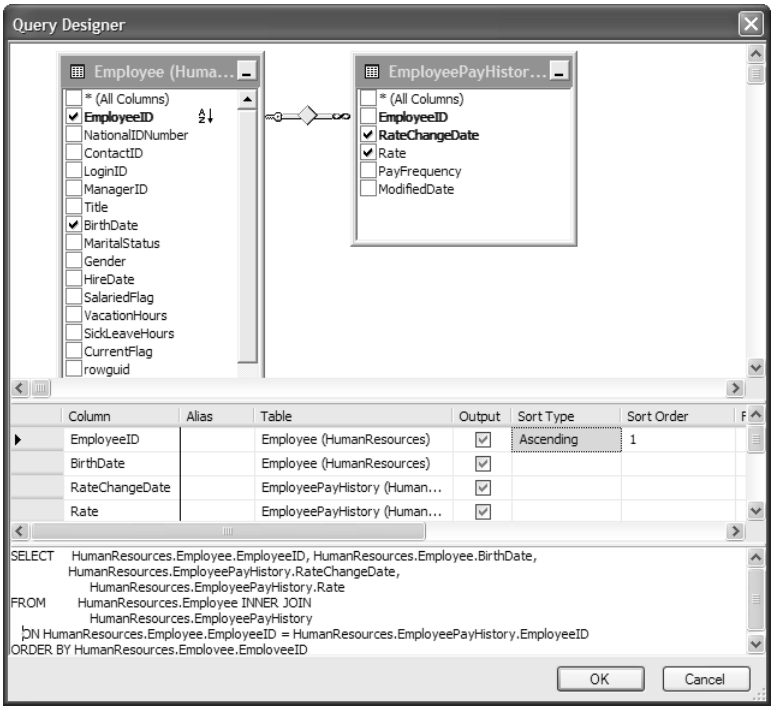

FIGURE 3.16 Designing queries in the Query Editor.

When you click OK on the Query Designer window, the related T-SQL is automatically placed in the Query Editor window. You can edit the T-SQL as needed or use it as is. You can imagine the time savings you can achieve by using this tool.

TIP

The Query Designer has a very impressive feature that allows you to view a T-SQL query visually. If you copy a valid T-SQL statement, open the Query Designer, and paste the T-SQL into the SQL pane at the bottom of the Query Designer, it tries to resolve the T-SQL into a graphical display. The tables in the `FROM` clause are shown in the designer panel, and information related to the selected columns is listed as well. The Query Designer cannot resolve all T-SQL statements and may fail to generate a visual display for some complex T-SQL.

Managing Projects in SSMS

Project management capabilities like those available in Visual Studio are now available in SSMS. Queries, connections, and other files that are related can be grouped into projects. A project or set of projects is further organized or grouped as a solution. This type of organization is the same as in the Visual Studio environment.

Projects and solutions are maintained and displayed with the Solution Explorer. The Solution Explorer contains a tree-like structure that organizes the projects and files in the solution. It is a component window within SSMS that you launch by selecting View, Solution Explorer. Figure 3.17 shows an example of the Solution Explorer. The solution in this example is named `EmployeeUpgrade`, and it contains two projects, named `Phase1` and `Phase2`. Each project contains a set of connections, a set of T-SQL scripts, and a set of miscellaneous files.

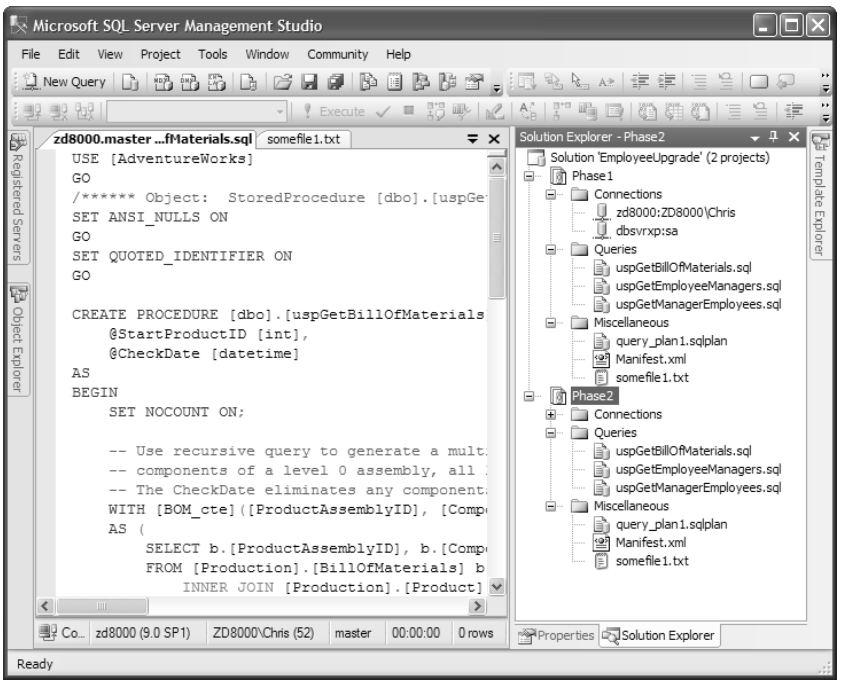

FIGURE 3.17 Solutions and projects listed in the Solution Explorer.

The first thing to do when using the project management capabilities in SSMS is to add a project. To do this, you select File, New, and when the New dialog appears, you select project to add a new project. It is a bit odd, but you must create the project before you can create the solution. When adding the new project, you are given a choice of the type of project, and you must select either SQL Server Scripts, Analysis Services Scripts, or SQL Mobile Scripts. Each one of these project types is geared toward the respective SQL Server technology.

After the project is added, you can add the related connections and files. To add a new connection, you simply right-click the `Connections` node. The `Connections` entries allow you to store SQL Server connection information that relates to the project you are working on. For example, you could have a connection to your test environment and

another connection to the production environment that relates to the project. When a connection is included in the project, you can double-click it, and a new query window for that connection is established.

SQL script files are added to a project in a similar fashion to connections: You right-click the `Queries` node and select the New Query option. A new Query Editor window appears, allowing you to enter the T-SQL commands. Any T-SQL script is viable for this category, including those that relate to database objects such as stored procedures, triggers, and tables.

You can also add existing files to a project. To do this, you right-click the project node, select Add, and then select Existing Item. The file types listed in the drop-down at the bottom of the Add Existing Item dialog include SQL Server files (`*.sql`), SQL deadlock files (`*.xdl`), XML files (`*.xml`), and execution plan files (`*.sqlplan`). SQL Server files are added, by default, to the `Queries` node. All the other file types are added to the `Miscellaneous` node. The connection entries are not stored in a separate file but are contained in the project file itself.

Integrating SSMS with Source Control

SSMS has the capability to integrate database project files into a source control solution. Source control provides a means for protecting and managing files. Source control applications typically contain features that allow you to track changes to files, control and track who uses the files, and provide a means for tagging the files with a version stamp so that the files can be retrieved at a later time, by version.

SSMS can integrate with a number of different source control applications. Visual SourceSafe is Microsoft's basic source control solution, but other source control applications can be used instead. The source control client application must be installed on the machine on which SSMS is running. When this is complete, you can set the source control application that SSMS will use within SSMS. To do this, you select Tools, Options and navigate to the `Source Control` node. The available source control clients are listed in the Current Source Control Plug-in drop-down.

The link between SSMS and the source control application is the database solution. After a solution has been created, it can be added to source control. To add a solution to a source control application, you open the Solution Explorer and right-click the solution or any of the projects in the solution. You then see the Add Solution to Source Control option. You must then log in to the source control application and select a source control project to add the solution to.

When the solution has been added to a source control application, all the related projects and project files are added as well. The projects and files that are in the source control application have additional options available in the Solution Explorer. Figure 3.18 shows a sample solution that has been added to a source control application. A subset of the source control options that are available when right-clicking project files are shown in this figure as well.

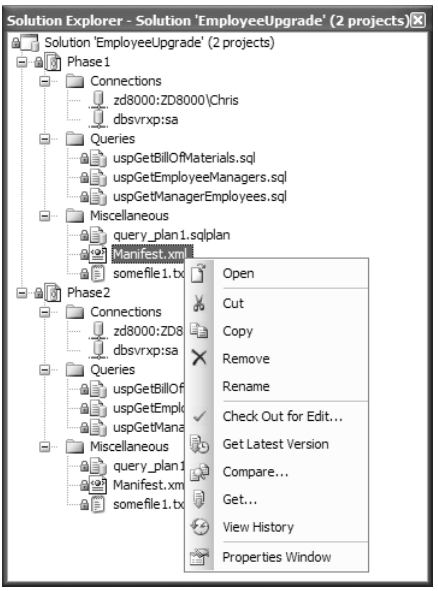

FIGURE 3.18 Source control options in the Solution Explorer.

The options that relate to source control are listed toward the bottom of the options list. The options that are available depend on the status of the selected file. For example, if a file has been checked out, additional options are displayed that relate to checking the file back in. The following are some of the common source control options:

- **Check Out for Edit**—This option allows you get a copy of the file from the source control application so that you can modify the file. When you check out the file, the source control provider can keep track of the user who has checked out the file, and it can also prevent other users from checking the file out.
- **Check In**—This option copies the locally modified file into the source control solution. The file must first be checked out for editing before you can use the Check In option. A new version for the file is established, and any prior versions of the file are retained as well.
- **Get Latest Version**—This option gets a read-only copy of the latest version of the project file from the source control application. The file is not checked out with this option.
- **Compare**—This option enables you to compare version of source control files. The default comparison that is shown is between the file in the source control application and the local file on your machine.
- **Get**—This option is similar to the Get Latest Version option, but it retrieves a read-only copy of the file. With this option, a dialog box appears, allowing you to select the file(s) that you want to retrieve.

- **View History**—This option lists all versions of the files that have been checked into the source control application. The History dialog box has many options that you can use with the different versions of the file. You can view differences between versions of the files, view the contents of a specific version, generate reports, or get an older version of the file.
- **Undo Checkout**—This option changes the checkout status in the source control application and releases the file to other source control users. Any changes that were made to the local copy of the file are not added to the source control version.

Other source control options are available via the Source Control menu in SSMS. You select an item in the Solution Explorer and then select File, Source Control. You can use this menu to check the status of a file by using the SourceSafe Properties option, set source control properties, launch the source control application, and perform other source control operations.

Using SSMS Templates

Templates provide a framework for the creation of database objects in SSMS. They are essentially boilerplate files that help generate scripts for common database objects. They can speed up the development of these scripts and help enforce consistency in the generation of the underlying database objects.

SQL Server 2005 has expanded the features available for generating templates. One substantial change is the addition of the Template Explorer. The Template Explorer is a component window that is available in SSMS and replaces the Template tab that was available in the SQL Server 2000 Query Analyzer. Figure 3.19 shows the Template Explorer and the available SQL Server template folders. Separate templates also exist for Analysis Services and SQL Server Mobile Edition. You can view them by selecting the related icon at the top of the Template Explorer.

You access the available templates by expanding the template folder in the Template Explorer tree. For example, if you expand the `Index` folder, you see six different types of index templates. If you double-click one of the templates, a new Query Editor window appears, populated with the template script. Figure 3.20 shows the template script that is displayed when you open the Create Index Basic template.

The template script contains template parameters that have the following format within the script:

```
<parameter_name, data_type, value>.
```

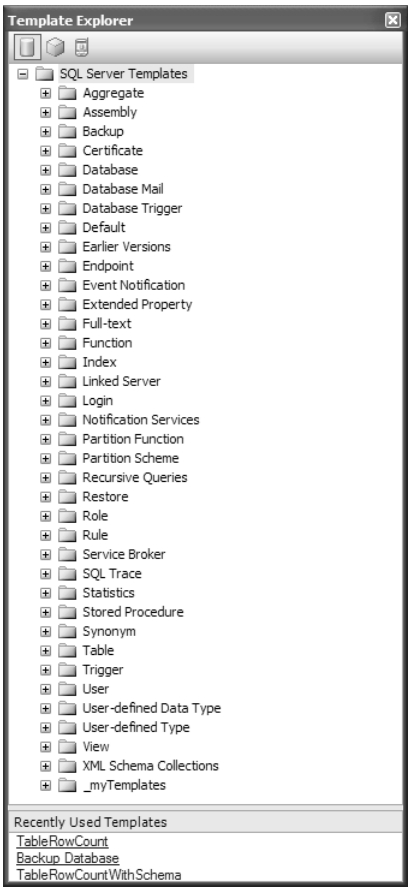

FIGURE 3.19 The SSMS Template Explorer.

You can manually replace these parameters in the script, or you can use the Specify Values for Template Parameters option from the Query menu to globally replace the parameters in the script with the desired values. Selecting Query, Specify Values for Template Parameters launches the Specify Values for Template Parameters dialog box, which enables you to enter the parameter values (see Figure 3.21).

TIP

When you use the Specify Values for Template Parameters option, some parameters may be missed if the parameter text has been altered. For example, if you add a carriage return after *parameter_name*, the Parameters dialog box does not list that parameter. It is best to leave the template script unchanged before you specify values for the parameters. You should make changes to the script after the values have been specified.

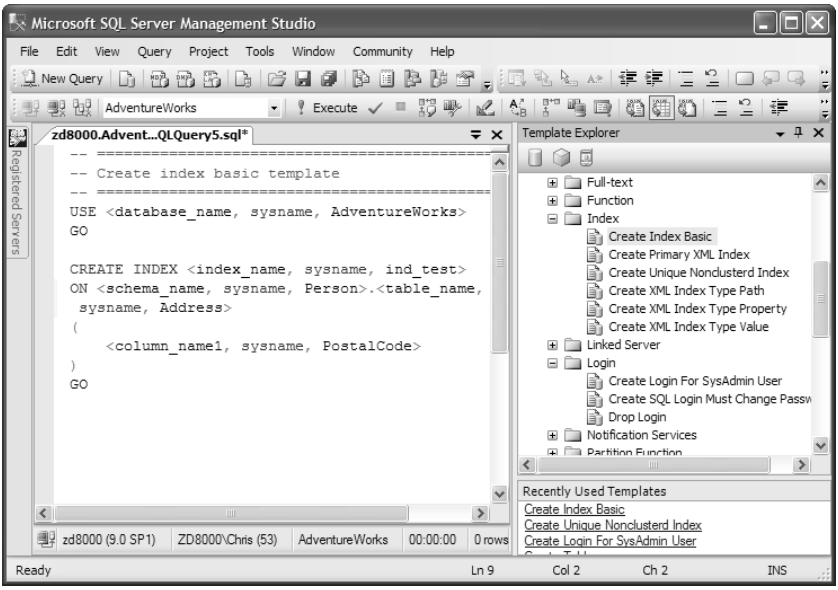

FIGURE 3.20 The template script for creating a basic index.

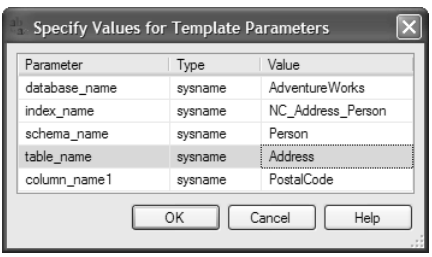

FIGURE 3.21 The Specify Values for Template Parameters dialog box.

After you have entered the parameter values and clicked OK, the values are reflected in the script. For example, the values shown in Figure 3.21 for the basic index template result in the following script:

```
-- ============================================
-- Create index basic template
-- ============================================
USE AdventureWorks
GO
CREATE INDEX NC_Address_Person
ON Person.Address
(
    PostalCode
)
GO
```

You also have the option of creating your own custom templates. These templates can contain parameters just like those that are available with the default templates. You can also create your own template folder that will be displayed in the Template Explorer tree. To create a new template folder, you right-click the `SQL Server Templates` node in the Template Explorer tree and select New, Folder. A new folder appears in the tree, and you can specify a new folder name. Figure 3.22 shows the Template Explorer with a set of custom templates found under the `_mytemplates` folder. The code pane in this figure shows the contents of a new custom template named `sys.objectSelectWithParameters`. This custom template contains two parameter declarations: `object_type` and `modify_date`. When you select the Specify Values for Template Parameters options for this custom template, you have the opportunity to change the values, just as you can with the default templates.

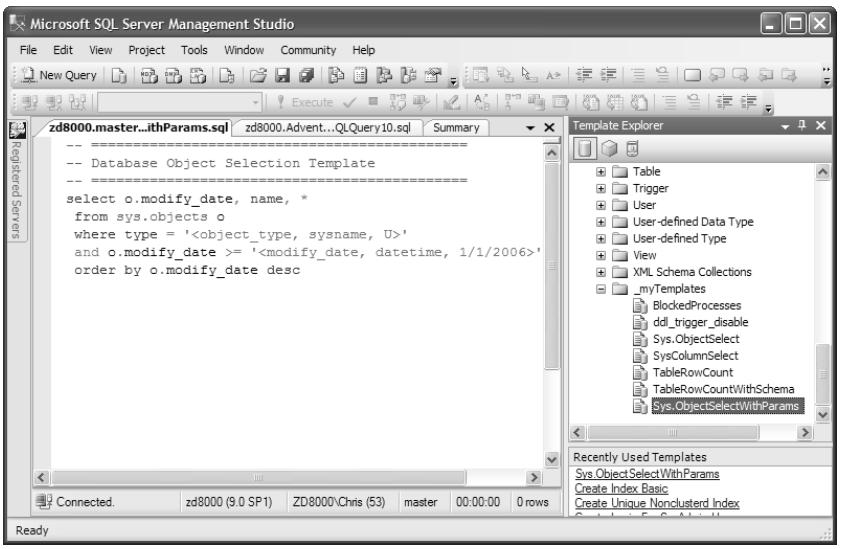

FIGURE 3.22 A custom template example.

NOTE

When you double-click a template in the Template Explorer tree, you create a script that is based on the template. Changes made to the script do not affect the template; they affect only the script that was generated from the template. To change the actual template, you need to right-click the template and select Edit. When you have completed your changes, you need to make sure to save the template.

Also, keep in mind that there is no requirement to have parameters in your templates. Templates are handy tools for accessing any code snippet you might use. After the code snippet is added as a template, you can open a new Query Editor window based on the template or simply drag and drop the template from the Template Explorer to an existing Query Editor window, and the code for the template is pasted into the window.

Summary

The number of tools and features available in SSMS is extensive and can be daunting when you first enter the environment. Remember that you can customize this environment and hide many of the windows that are displayed. You can start with a fairly simple SSMS configuration that includes the Object Explorer and a Query Editor window. This allows you to accomplish a majority of your SQL Server tasks. As you become more familiar with the environment, you can introduce new tools and features to help improve your overall productivity.

The discussion of SSMS does not end with this chapter. Further details related to SSMS are covered throughout this book. You can use the new features described in this chapter as a starting point and look to other chapters for more detailed discussion of database features that are accessible through SSMS.

Chapter 4 looks at the SQL Server utilities that can be run from the command prompt. These tools allow you to perform some of the same tasks that are available in SSMS. The ability to launch these utilities from the command line can be useful when you're automating tasks or accessing SQL Server when graphical user interface tools such as SSMS are not available.

This page intentionally left blank

CHAPTER 4

SQL Server Command-Line Utilities

IN THIS CHAPTER

This chapter explores various command-line utilities that ship with SQL Server. These utilities give administrators a different way to access the database engine and its related components. In some cases, they provide functionality that is also available with SQL Server's graphical user interface (GUI). Other command-line utilities provide functionality that is available only from the command prompt. For each utility, this chapter provides the command syntax along with the most commonly used options. For the full syntax and options available for the utility, see SQL Server Books Online.

NOTE

The focus of this chapter is on command-line utilities that are core to SQL Server and the SQL Server database engine. Several other command-line utilities, geared toward other SQL Server services, are not covered in this chapter. These utilities include `dtexec` and `dtutil`, which can be used with SQL Server Integration Services (SSIS). Reporting Services has the `rs`, `rsconfig`, and `rskeymgmt` command-line utilities, and Notification Services has the `nscontrol` utility. These utilities are beyond the scope of this chapter, but they are worth mentioning.

Table 4.1 lists the command-line utilities that are discussed in this chapter. This table lists the physical location of each utility's executable. The location is needed to execute the utility in most cases, unless the associated path has been added to the `Path` environmental variable.

TABLE 4.1 Command-Line Utility Installation Locations

Utility	Install Location
sqlcmd	x:\Program Files\Microsoft SQL Server\90\Tools\Binn
dta	x:\Program Files\Microsoft SQL Server\90\Tools\Binn
tablediff	x:\Program Files\Microsoft SQL Server\90\COM
sac	x:\Program Files\Microsoft SQL Server\90\Shared
bcp	x:\Program Files\Microsoft SQL Server\90\Tools\Binn
sqldiag	x:\Program Files\Microsoft SQL Server\90\Tools\Binn
sqlservr	x:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Binn

When testing many of these utilities, it is often easiest to set up a batch file (`.BAT`) that contains a command to change the directory to the location shown in Table 4.1. After you make this directory change, you can enter the command-line utility with the relevant parameters. Finally, you should enter a `PAUSE` command so that you can view the output of the utility in the command-prompt window. The following is an example that can be used to test the `sac` utility (which is discussed in more detail later in this chapter):

```
cd "C:\Program Files\Microsoft SQL Server\90\Shared
sac out c:\SAC_Features_output.xml -I MSSQLSERVER -F
pause
```

After you save the commands in a file with a `.BAT` extension, you can simply double-click the file to execute it. This is much easier than retyping the commands many times during the testing process.

What's New in SQL Server Command-Line Utilities

SQL Server 2005 offers a number of new command-line utilities that augment the capabilities available with the SQL Server 2005 graphical tools and in some cases provide functionality that is available only from the command prompt. The following new tools are discussed in detail later in this chapter:

- **`sqlcmd`**—The `sqlcmd` utility allows you to execute Transact-SQL (T-SQL) statements and scripts from the command prompt. It provides the same type of functionality that was available in `isql` and `osql` in previous versions of SQL Server, but it offers a number of new script execution options that go beyond what was available before.
- **`dta`**—The graphical Database Engine Tuning Advisor (DTA) has a related command-line utility named `dta`. `dta` is used to analyze a database workload and provide physical design recommendations that can be used to optimize performance.
- **`tablediff`**—This utility allows you to compare the data contained within two different tables. This utility was designed to help troubleshoot replication differences, but it can be used in many scenarios where differences in table data must be identified.

- **sac**—This utility can be used to import or export settings that related to surface area configuration. It can be a real timesaver and a means for ensuring consistency across a number of SQL Server installations.

Several other command-line utilities that are also new to SQL Server 2005 can be used to launch the graphical tools that come with SQL Server:

- **profiler90**—This utility launches the SQL Server Profiler application.
- **sqlwb**—This utility launches SQL Server Management Studio (SSMS).
- **dtswizard**—This utility launches the SQL Server Import/Export Wizard, which allows you to move data to and from SQL Server data sources. This is the same tool that is launched from SSMS by right-clicking a database and choosing Tasks, Export Data or Import Data.

These tools are not discussed in detail in this chapter, but they are handy alternatives to launching the GUI tools. You simply click Start, Run and enter the utility name or type in the utility name at a command prompt and press Enter.

The sqlcmd Command-Line Utility

The sqlcmd command-line utility is the next generation of the isql and osql utilities that you may have used in prior versions of SQL Server. It provides the same type of functionality as isql and osql, including the ability to connect to SQL Server from the command prompt and execute T-SQL commands. The T-SQL commands can be stored in a script file, entered interactively, or specified as command-line arguments to sqlcmd.

NOTE

The isql and osql command-line utilities are not covered in this chapter. isql is no longer supported in SQL Server 2005. The osql utility is still supported but will be removed in a future version of SQL Server. Make sure to use sqlcmd in place of isql or osql.

The syntax for sqlcmd follows:

```
sqlcmd
[{ { -U login_id [ -P password ] } ¦ -E trusted connection }]
[ -z new password ] [ -Z new password and exit]
[ -S server_name [ \ instance_name ] ] [ -H wksta_name ] [ -d db_name ]
[ -l login time_out ] [ -A dedicated admin connection ]
[ -i input_file ] [ -o output_file ]
[ -f < codepage > ¦ i: < codepage > [ < , o: < codepage > ] ]
[ -u unicode output ] [ -r [ 0 ¦ 1 ] msgs to stderr ]
[ -R use client regional settings ]
[ -q "cmdline query" ] [ -Q "cmdline query" and exit ]
```

```
[ -e echo input ] [ -t query time_out ]
[ -I enable Quoted Identifiers ]
[ -v var = "value"...] [ -x disable variable substitution ]
[ -h headers ][ -s col_separator ] [ -w column_width ]
[ -W remove trailing spaces ]
[ -k [ 1 ¦ 2 ] remove[replace] control characters ]
[ -y display_width ] [-Y display_width ]
[ -b on error batch abort ] [ -V severitylevel ] [ -m error_level ]
[ -a packet_size ][ -c cmd_end ]
[ -L [ c ] list servers[clean output] ]
[ -p [ 1 ] print statistics[colon format]]
[ -X [ 1 ] ] disable commands, startup script, environment variables [and exit]
[ -? show syntax summary ]
```

The number of options available for `sqlcmd` is extensive, but many of the options are not necessary for basic operations. To demonstrate the usefulness of this tool, we will look at several different examples of the `sqlcmd` utility, from fairly simple (using few options) to more extensive.

Executing the `sqlcmd` utility

Before getting into the examples, it is important to remember that `sqlcmd` can be run in several different ways. It can be run interactively from the command prompt, from a batch file, or from a Query Editor window in SSMS. When run interactively, the `sqlcmd` program name is entered at the command prompt with the required options to connect to the database server. When the connection is established, a numbered row is made available to enter the T-SQL commands. Multiple rows of T-SQL can be entered in a batch; they are executed only after the `GO` command has been entered. Figure 4.1 shows an example with two simple `SELECT` statements that were executed interactively with `sqlcmd`. The connection in this example was established by simply typing `sqlcmd` at the command prompt to establish a trusted connection to the default instance of SQL Server running on the machine on which the command prompt window is opened.

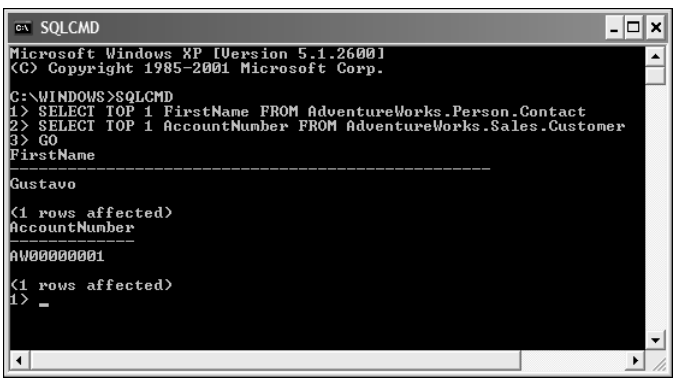

FIGURE 4.1 Executing `sqlcmd` interactively.

The ability to edit and execute sqlcmd scripts within SSMS is new to SQL Server 2005. A sqlcmd script can be opened or created in a Query Editor window within SSMS. To edit these scripts, you must place the editor in sqlcmd mode. You do so by selecting Query, sqlcmd Mode or by clicking the related toolbar button. When the editor is put in sqlcmd mode, it provides color coding and the ability to parse and execute the commands within the script. Figure 4.2 shows a sample sqlcmd script that is opened in SSMS in a Query Editor window that has been set to sqlcmd mode. The shaded lines are sqlcmd commands.

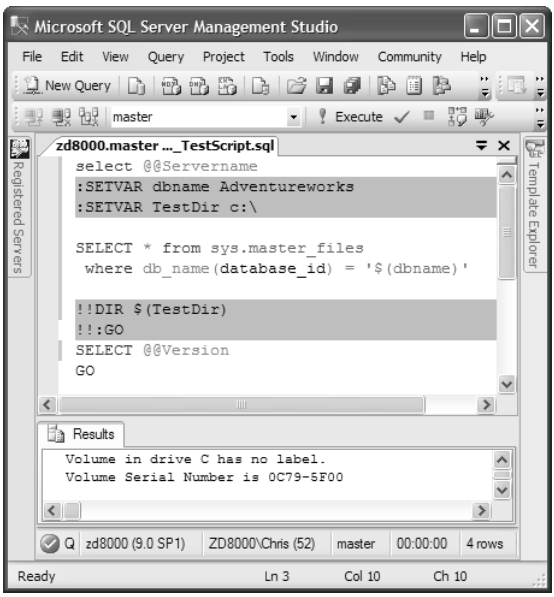

FIGURE 4.2 Executing and editing sqlcmd scripts in SSMS.

The most common means for executing sqlcmd utility is via a batch file. This method can provide a great deal of automation because it allows you to execute a script or many scripts by launching a single file. The examples shown in this section are geared toward the execution of sqlcmd in this manor. The following example illustrates the execution of sqlcmd, using a trusted connection to connect to the local database, and the execution of a simple query that is set using the -Q option:

```
sqlcmd -S (local) -E -Q"select getdate()"
```

You can expand this example by adding an output file to store the results of the query and add the -e option, which echoes the query that was run in the output results:

```
sqlcmd -S (local) -E -Q"select getdate()" -o c:\TestOutput.txt -e
```

The contents of the c:\TestOutput.txt file should look similar to this:

```
select getdate()
----------------------
2006-09-13 20:27:25.343
 (1 rows affected)
```

Using a trusted connection is not the only way to use sqlcmd to connect to a SQL Server instance. You can use the -U and -P command-line options to specify the SQL Server user and password to use to connect to the server. sqlcmd also provides an option to specify the password in an environmental variable named sqlcmdPASSWORD, which can be assigned prior to the sqlcmd execution and eliminates the need to hard-code the password in a batch file.

sqlcmd also provides a means for establishing a dedicated administrator connection (DAC) to the server. The DAC, which is new to SQL Server 2005, is typically used for troubleshooting on a server that is having problems. It allows an administrator to get onto the server when others may not be able to. If the DAC is enabled on the server, a connection can be established with the -A option and a query can be run, as shown in the following example:

```
sqlcmd -S (local) -A -Q"select getdate()"
```

To manage more complex T-SQL execution, it is typically easier to store the T-SQL in a separate input file. The input file can then be referenced as a sqlcmd parameter. For example, let's say that you have the following T-SQL stored in a file named C:\TestsqlcmdInput.sql:

```
BACKUP DATABASE Master
 TO DISK = 'c:\master.bak'

BACKUP DATABASE Model
 TO DISK = 'c:\model.bak'

BACKUP DATABASE MSDB
 TO DISK = 'c:\msdb.bak'
```

The sqlcmd execution, which accepts the C:\TestsqlcmdInput.sql file as input and executes the commands within the file, looks like this:

```
sqlcmd -S (local) -E -i"C:\TestsqlcmdInput.sql" -o c:\TestOutput.txt -e
```

The execution of the previous example backs up three of the system databases and writes the results to the output file specified.

Using scripting variables with sqlcmd

sqlcmd provides a means for utilizing variables within sqlcmd input files or scripts. These scripting variables can be assigned as sqlcmd parameters or set within the sqlcmd script.

To illustrate the use of scripting variables, let's change our previous backup example so that the database that is going to be backed up is a variable. A new input file named `c:\BackupDatabase.sql` should be created, and it should contain the following command:

```
BACKUP DATABASE $(DatabaseToBackup)
 TO DISK = 'c:\$(DatabaseToBackup).bak'
```

The variable in the previous example is named `DatabaseToBackup`. Scripting variables are referenced using the `$( )` designators. These are resolved at the time of execution, and a simple replacement is performed. This allows variables to be specified within quotes, if necessary. The `-v` option is used to assign a value to a variable at the command prompt, as shown in the following example, which backs up the `model` database:

```
sqlcmd -S (local) -E -i"C:\BackupDatabase.sql" -v DatabaseToBackup = model
```


If multiple variables exist in the script, they can all be assigned after the `-v` parameter and should not be separated by a delimiter, such as a comma or semicolon. Scripting variables can also be assigned within the script, using the `:SETVAR` command. The input file from the previous backup would be modified as follows to assign the `DatabaseToBackup` variable within the script:

```
:SETVAR DatabaseToBackup Model
BACKUP DATABASE $(DatabaseToBackup)
 TO DISK = 'c:\$(DatabaseToBackup).bak'
```

Scripts that utilize variables, `sqlcmd` commands, and the many available options can be very sophisticated and can make your administrative life easier. The examples in this section illustrate some of the basic features of `sqlcmd`, including some of the new features in SQL Server 2005 that go beyond what was available in ISQL or OSQL.

The `dta` Command-Line Utility

`dta` is the command-line version of the graphical Database Engine Tuning Advisor. They both provide performance recommendations based on the workload provided to them. The syntax for `dta` is as follows:

```
            Dta [ -? ] ¦
    [
         [ -S server_name[ \instance ] ]
         {
             { -U login_id [-P password ] }
             ¦ –E             }
             { -D database_name [ ,...n ] }
                 [-d database_name ]
                 [ -Tl table_list ¦ -Tf table_list_file ]
             { -if workload_file ¦ -it workload_trace_table_name }
```

```
            { -s session_name ¦ -ID session_ID }
                [ -F ]
                        [ -of output_script_file_name ]
                        [ -or output_xml_report_file_name ]
                        [ -ox output_XML_file_name ]
                        [ -rl analysis_report_list [ ,...n ] ]
                        [ -ix input_XML_file_name ]
                        [ -A time_for_tuning_in_minutes ]
                        [ -n number_of_events ]
                [ -m minimum_improvement ]
                        [ -fa physical_design_structures_to_add ]
                        [ -fp partitioning_strategy ]
                        [ -fk keep_existing_option ]
                        [ -fx drop_only_mode ]
                [ -B storage_size ]
                [ -c max_key_columns_in_index ]
                [ -C max_columns_in_index ]
                        [ -e ¦ -e tuning_log_name ]
                        [ -N online_option]
                        [ -q ]
                     [ -u ]
                    [ -x ]
                    [ -a ]
]
```

There are an extensive number of options available with this utility, but many of them are not required to do basic analysis. At a minimum, you need to use options that provide connection information to the database, a workload to tune, a tuning session identifier, and the location to store the tuning recommendations. The connection options include `-S` for the server name, `-D` for the database, and either `-E` for a trusted connection or `-U` and `-P`, which can be used to specify the user and password.

The workload to tune is either a workload file or a workload table. The `-if` option is used to specify the workload file location, and the `-it` option is used to specify a workload table. The workload file must be a Profiler trace file (`.trc`), a SQL script (`.sql`) that contains T-SQL commands, or a SQL Server trace file (`.log`). The workload table is a table that contains output from a workload trace. The table is specified in the form *`database_name.owner_name.table_name`*.

The tuning session must be identified with either a session name or a session ID. The session name is character based and is specified with the `-s` option. If the session name is not provided, a session ID must be provided instead. The session ID is numeric and is set using the `-ID` option. If the session name is specified instead of the session ID, the `dta` generates an ID anyway.

The last options that are required for a basic `dta` execution identify the destination to store the `dta` performance recommendations. The performance recommendations can be

stored in a script file or in XML. The `-of` option is used to specify the output script filename. XML output is generated when the `-or` or `-ox` options are used. The `-or` option generates a filename if one is not specified, and the `-ox` option requires a filename. The `-F` option can be used with any of the output options to force an overwrite of a file with the same name, if one exists.

To illustrate the use `dta` with the basic options, let's look at an example of tuning a simple `SELECT` statement against the `AdventureWorks` database. To begin, you use the following T-SQL, which is stored in a workload file named `c:\myScript`:

```
USE AdventureWorks ;
GO
select *
 from Production.transactionHistory
 where TransactionDate = '9/1/04'
```


The following example shows the basic `dta` execution options that can be used to acquire performance recommendations:

```
dta -S zd8000 -E -D AdventureWorks -if c:\MyScript.sql
-s MySessionX -of C:\MySessionOutputScript.sql -F
```

NOTE

`dta` and other utilities that are executed at the command prompt are executed with all the options on a single line. The previous example and any others in this chapter that are displayed on more than one line should actually be executed at the command prompt or in a batch file on a single line. They are broken here only because the printed page can only accommodate a fixed number of characters.

The previous example utilizes a trusted connection against the `AdventureWorks` database, a workload file named `c:\MyScript.sql`, and a session named `MySessionX`, and it outputs the performance recommendations to a text file named `c:\MySessionOutputScript.sql`. The `-F` option is used to force a replacement of the output file if it already exists. The output file contains the following performance recommendations:

```
use [AdventureWorks]
go

CREATE NONCLUSTERED INDEX [_dta_index_TransactionHistory_6]
 ON [Production].[TransactionHistory]
(
    [TransactionDate] ASC
)
INCLUDE ( [TransactionID],
[ProductID],
[ReferenceOrderID],
```

```
[ReferenceOrderLineID],
[TransactionType],
[Quantity],
[ActualCost],
[ModifiedDate]) WITH (SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, IGNORE_DUP_KEY = OFF,
ONLINE = OFF) ON [PRIMARY]
go
```

In short, the `dta` output recommends that a new index be created on the `TransactionDate` column in the `TransactionHistory` table. This is a viable recommendation, considering that there was no index on the `TransactionHistory.TransactionDate` column, and it was used as a search argument in the workload file.

Many other options (that go beyond basic execution) can be used to manipulate the way `dta` makes recommendations. For example, a list can be provided to limit which tables the `dta` looks at during the tuning process. Options can be set to limit the amount of time that the `dta` tunes or the number of events. These options go beyond the scope of this chapter, but you can gain further insight about them by looking at the graphical DTA, which contains many of the same types of options. You can refine your tuning options in the DTA, export the options to an XML file, and use the `-ix` option with the `dta` utility to import the XML options and run the analysis.

The `tablediff` Command-Line Utility

The `tablediff` utility is a new addition to SQL Server 2005. This utility enables you to compare the contents of two tables. It was originally developed for replication scenarios to help troubleshoot nonconvergence, but it is also very useful in other scenarios. When data in two tables should be the same or similar, this tool can help determine whether they are the same, and if they are different, it can identify what data in the tables is different.

The syntax for `tablediff` is as follows:

```
tablediff
[ -? ] ¦
{
        -sourceserver source_server_name[\instance_name]
        -sourcedatabase source_database
        -sourcetable source_table_name
    [ -sourceschema source_schema_name ]
    [ -sourcepassword source_password ]
    [ -sourceuser source_login ]
    [ -sourcelocked ]
        -destinationserver destination_server_name[\instance_name]
        -destinationdatabase subscription_database
        -destinationtable destination_table
```

```
    [ -destinationschema destination_schema_name ]
    [ -destinationpassword destination_password ]
    [ -destinationuser destination_login ]
    [ -destinationlocked ]
    [ -b large_object_bytes ]
    [ -bf number_of_statements ]
    [ -c ]
    [ -dt ]
    [ -et table_name ]
    [ -f [ file_name ] ]
    [ -o output_file_name ]
    [ -q ]
    [ -rc number_of_retries ]
    [ -ri retry_interval ]
    [ -strict ]
    [ -t connection_timeouts ]
}
```


The `tablediff` syntax requires source and destination connection information in order to perform a comparison. This information includes the servers, databases, and tables that will be compared. Connection information must be provided for SQL Server authentication but can be left out if Windows authentication can be used. The source and destination parameters can be for two different servers or the same server, and the `tablediff` utility can be run on a machine that is neither the source nor the destination.

To illustrate the usefulness of this tool, let's look at a sample comparison in the `AdventureWorks` database. The simplest way to create some data for comparison is to select the contents of one table into another and then update some of the rows in one of the tables. The following `SELECT` statement makes a copy of the `AddressType` table in the `AdventureWorks` database to the `AddressTypeCopy` table:

```
select *
 into Person.AddressTypeCopy
 from Person.AddressType
```

In addition, the following statement updates two rows in the `AddressTypeCopy` table so that you can use the `tablediff` utility to identify the changes:

```
UPDATE Person.AddressTypeCopy
 SET Name = 'Billing New'
 WHERE AddressTypeId = 1

UPDATE Person.AddressTypeCopy
 SET Name = 'Shipping New',
  ModifiedDate = '20060918'
 WHERE AddressTypeId = 5
```

The `tablediff` utility can be executed with the following parameters to identify the differences in the `AddressType` and `AddressTypeCopy` tables:

```
tablediff -sourceserver "(local)" -sourcedatabase "AdventureWorks"
-sourceschema "Person"-sourcetable "AddressType"
-destinationserver "(local)" -destinationdatabase "AdventureWorks"
-destinationschema "Person" -destinationtable "AddressTypeCopy"
-f c:\TableDiff_Output.txt
```

The destination and source parameters are the same as in the previous example, except for the table parameters, which have the source `AddressType` and the destination `AddressTypeCopy`. The execution of the utility with these parameters results in the following output to the command prompt window:

```
User-specified agent parameter values:
-sourceserver (local)
-sourcedatabase AdventureWorks
-sourceschema Person
-sourcetable AddressType
-destinationserver (local)
-destinationdatabase AdventureWorks
-destinationschema Person
-destinationtable AddressTypeCopy
-f c:\TableDiff_Output

Table [AdventureWorks].[Person].[AddressType] on (local)
and Table [AdventureWorks].[Person].[AddressTypeCopy] on (local)
have 2 differences.
Fix SQL written to c:\TableDiff_Output.sql.
Err     AddressTypeID   Col
Mismatch        1       Name
Mismatch        5       ModifiedDate Name
The requested operation took 0.296875 seconds.
```

The output first displays a summary of the parameters used and then shows the comparison results. In this example, it found the two differences that are due to updates that were performed on `AddressTypeCopy`. In addition, the `-f` parameter that was used in the example caused the `tablediff` utility to output a SQL file that can be used to fix the differences in the destination table. The output file from this example looks as follows:

```
-- Host: (local)
-- Database: [AdventureWorks]
-- Table: [Person].[AddressTypeCopy]
SET IDENTITY_INSERT [Person].[AddressTypeCopy] ON
UPDATE [Person].[AddressTypeCopy]
```

```
 SET [Name]='Billing'
 WHERE [AddressTypeID] = 1
UPDATE [Person].[AddressTypeCopy]
 SET [ModifiedDate]='1998-06-01 00:00:00.000',
 [Name]='Shipping' WHERE [AddressTypeID] = 5
SET IDENTITY_INSERT [Person].[AddressTypeCopy] OFF
```

NOTE

The `tablediff` utility requires the source table to have at least one primary key, identity, or `ROWGUID` column. This gives the utility a key that it can use to try to match a corresponding row in the destination table. If the `-strict` option is used, the destination table must also have a primary key, identity, or `ROWGUID` column.

Keep in mind that several different types of comparisons can be done with the `tablediff` utility. The `-q` option causes a quick comparison that compares only record counts and looks for differences in the schema. The `-strict` option forces the schemas of each table to be the same when the comparison is run. If this option is not used, the utility allows some columns to be of different data types, as long as they meet the mapping requirements for the data type (for example, `INT` can be compared to `BIGINT`).

The `tablediff` utility can be used for many different types of comparisons. How you use this tool depends on several factors, including the amount and type of data you are comparing.

The `sac` Command-Line Utility

The `sac` utility allows you to import or export settings that are also available in the GUI Surface Area Configuration (SAC) tool. Typically, you define these settings with the GUI tool, and then you can use the `sac` utility to export those settings to a file. This file can then be deployed to other machines so that you end up with a consistent set of `sac` settings.

The syntax for the `sac` utility follows:

```
sac {in ¦ out} filename [-S computer_name]
   [-U SQL_login [-P SQL_ password]]
   [-I instance_name ]
   [-DE] [-AS] [-RS] [-IS] [-NS] [-AG] [-BS] [-FT] [-AD]
   [-F] [-N] [-T] [-O]
   [-H ¦ -?]
```

Table 4.2 gives a brief explanation of each of the `sac` parameters.

TABLE 4.2 sac Parameters

Option	Description
in	Used to import settings.
out	Used to export settings.
-S	Specifies the target computer name.
-U	Specifies the SQL Server login. If this option is not used, Windows authentication is used.
-P	Specifies the SQL Server password associated with the login set with the –U parameter.
-I	Specifies the name of the SQL Server instance to connect to. If it is not specified, all instances are targeted.
-DE	Used to import or export database engine settings.
-AS	Used to import or export Analysis Services settings.
-RS	Used to import or export Reporting Services settings.
-IS	Used to import or export SSIS settings.
-NS	Used to import or export Notification Services settings.
-AG	Used to import or export SQL Server Agent settings.
-BS	Used to import or export SQL Server Browser settings.
-FT	Used to import or export Full-Text Search settings.
-F	Used to import or export feature settings.
-N	Used to import or export network protocols for remote connectivity.
-T	Used to import or export service settings for SQL Server components.
-O	Used to set the name of the file that the command-line output will be written to.
-H -?	Used to get help on sac syntax.

The –S, –U and –P, and –I parameters define the connection information to the target or destination server. Most of the remaining options determine which settings will be imported or exported. These settings are broken down into two main categories: Surface Area Configuration for Services and Connections and Surface Area Configuration for Features. These same two categories are also displayed in the SQL Server 2005 SAC GUI application, as shown at the bottom of Figure 4.3.

When the –F option is used with the sac utility, all the feature options are imported or exported. These are the same features that can be defined with the Surface Area Configuration for Features selection in the GUI application. These features bridge several different components and are either enabled or disabled:

- **Database engine features**—Database engine features include ad hoc remote queries, CLR integration, DAC, Database Mail, Native XML Web services, OLE automation, Service Broker, SQL Mail, Web Assistant, and Xp_cmdshell.
- **Analysis Services features**—Analysis Services features include ad hoc mining queries, anonymous connections, linked objects, and user-defined functions.
- **Reporting Services features**—Reporting Services features include scheduled event and report delivery, Web service and HTTP access, and Windows integrated security.

FIGURE 4.3 Configuration options on the SAC GUI.

The following is a sample execution of the `sac` utility, which exports all the features for the default instance on the local computer on which the `sac` utility is run:

```
sac out c:\SAC_Features_output.xml -I MSSQLSERVER -F
```

The output file is in XML format and contains a named pair for each feature listed previously. You can transfer this same output file to another machine and execute it with the `in` clause in order to import the settings to that machine, as shown in the following example:

```
sac in c:\SAC_Features_output.xml -I MSSQLSERVER -F
```

The other category of settings that can be imported or exported with `sac` relate to services and connections. Each service (for example, the database engine or Integration Services) has a number of settings, including how it will start and the status of the service. In addition, the database engine and Analysis Services have settings that relate to the state of the network protocols for remote connectivity. You can isolate the service settings by using the `-T` parameter, and you can use the `-N` option to isolate the settings related to network protocols. The following example gets all settings related to services and connections:

```
sac out c:\SAC_ServicesAndConnections_output.xml -I MSSQLSERVER -T -N
```

If the sac utility is run without the -F parameter or any other setting-specific parameters, all the settings will be targeted, including all settings for features, services, and connections, as shown in the following example:

```
sac in c:\SAC_Features_output.xml -I MSSQLSERVER
```

Many of the remaining parameters are used to isolate settings for specific services or components. For example, the -DE parameter causes sac to focus only on the database engine settings. The -RS parameter causes sac to only import or export the settings related to Reporting Services. More than one of these types of parameters can be used on a single sac execution, as shown in the following example, which exports all settings for the database engine and the SQL Server Agent:

```
sac out c:\SAC_dbAndAgent_output.xml -I MSSQLSERVER -DE -AG
```

The sac parameters provide a great deal of flexibility in choosing the settings that you want to work with. After you determine which settings you want to isolate, you can export them and then import them in the database environments that you want to make consistent.

The bcp Command-Line Utility

You use the bcp (bulk copy program) tool to address the bulk movement of data. The utility is bidirectional, allowing for the movement of data into and out of a SQL Server database.

The SQL Server 2005 and SQL Server 2000 versions of bcp utilize the ODBC bulk copy API instead of the DB-LIB API that was used in earlier versions of SQL Server. The ODBC bulk copy API is used to support new data types that the DB-LIB API does not support. Backward compatibility options are provided with the SQL Server 2005 bcp utility to allow bulk copy of data types supported in earlier versions.

bcp uses the following syntax:

```
bcp {[[database_name.][owner].]{table_name ¦ view_name} ¦ "query"}
    {in ¦ out ¦ queryout ¦ format} data_file
    [-mmax_errors] [-fformat_file] [-x] [-eerr_file]
    [-Ffirst_row] [-Llast_row] [-bbatch_size]
    [-n] [-c] [-N] [-w] [-V (60 ¦ 65 ¦ 70 ¦ 80)] [-6]
    [-q] [-C { ACP ¦ OEM ¦ RAW ¦ code_page } ] [-tfield_term]
    [-rrow_term] [-iinput_file] [-ooutput_file] [-apacket_size]
    [-Sserver_name[\instance_name]] [-Ulogin_id] [-Ppassword]
    [-T] [-v] [-R] [-k] [-E] [-h"hint [,...n]"]
```

Some of the commonly used options—other than the ones used to specify the database, such as user ID, password, and so on—are the -F and -L options. These options allow you to specify the first and last row of data to be loaded from a file, which is especially helpful in large batches. The -t option allows you to specify the field terminator that separates

data elements in an ASCII file. The -E option allows you to import data into SQL Server fields that are defined with identity properties.

TIP

The BULK INSERT T-SQL statement and SSIS are good alternatives to bcp. The BULK INSERT statement is limited to loading data into SQL Server, but it is an extremely fast tool for loading data. SSIS is a sophisticated GUI that allows for both data import and data export, and it has capabilities that go well beyond those that were available in SQL Server 2000's Data Transformation Services (DTS).

This section barely scratches the surface when it comes to the capabilities of bcp. For a more detailed look at bcp, refer to the section, "Using bcp" in Chapter 40, "SQL Server Integration Services."

The sqldiag Command-Line Utility

sqldiag is a diagnostic tool that you can use to gather diagnostic information regarding various SQL Server services. It is intended for use by Microsoft support engineers, but you might also find the information it gathers useful in troubleshooting a problem. sqldiag collects the information into files that are written, by default, to a folder named SQLDIAG, where the file sqldiag.exe is located (for example, C:\Program Files\Microsoft SQL Server\90\Tools\binn\SQLDIAG\). The folder holds files that contain information about the machine on which SQL Server is running in addition to the following types of diagnostic information:

- SQL Server configuration information
- SQL Server blocking output
- SQL Server Profiler traces
- Windows performance logs
- Windows event logs

The syntax for sqldiag has changed quite a bit in SQL Server 2005, and some of the options that were used in prior versions of sqldiag are not compatible with the current version. The full syntax for sqldiag is as follows:

```
sqldiag
     { [/?] }
     ¦
     { [/I configuration_file]
       [/O output_folder_path]
       [/P support_folder_path]
       [/N output_folder_management_option]
       [/C file_compression_type]
```

```
    [/B [+]start_time]
    [/E [+]stop_time]
    [/A SQLdiag_application_name]
    [/Q] [/G] [/R] [/U] [/L] [/X] }
  ¦
{ [START ¦ STOP ¦ STOP_ABORT] }
  ¦
{ [START ¦ STOP ¦ STOP_ABORT] /A SQLdiag_application_name }
```

NOTE

Keep in mind that many of the new options for `sqldiag` identify how and when the `sqldiag` utility will be run. The utility can now be run as a service, scheduled to start and stop at a specific time of day, and it can be configured to change the way the output is generated. The details about these options are beyond the scope of this chapter but are covered in detail in SQL Server Books Online. This section is intended to give you a taste of the useful information that this utility can capture.

By default, the `sqldiag` utility must be run by a member of the Windows Administrators group, and this user must also be a member of the sysadmin fixed SQL Server role. To get a flavor for the type of information that `sqldiag` outputs, you should open a command prompt window, change the directory to the location of the `sqldiag.exe` file, and type the following command:

```
sqldiag
```

No parameters are needed in order to generate the output. The command prompt window scrolls status information across the screen as it collects the diagnostic information. You see the message "SQLDIAG Initialization starting..." followed by messages that indicate what information is being collected. The data collection includes a myriad of system information from `MSINFO32`, default traces, and SQLDumper log files. When you are ready to stop the collection, you can press Ctrl+C.

If you navigate to the `sqldiag` output folder, you find the files that were created during the collection process. You should find a file with a name containing `MSINFO32` in this output folder. It contains the same type of information that you see when you launch the System Information application from Accessories or when you run `MSINFO32.EXE`. This is key information about the machine on which SQL Server is running. This information includes the number of processors, the amount of memory, the amount of disk space, and a slew of other hardware and software data.

You also find a file named *xxx*`_sp_sqldiag_Shutdown.out`, where *xxx* is the name of the SQL Server machine. This file contains SQL Server–specific information, including the SQL Server error logs, output from several key system stored procedures, including `sp_helpdb` and `sp_configure`, and much more information related to the current state of SQL Server.

You find other files in the sqldiag output directory as well. Default trace files, log files related to the latest sqldiag execution, and a copy of the XML file containing configuration information are some of them. Microsoft documentation on these files is limited, and you may find that the best way to determine what they contain is simply to open the files and review the wealth of information they contain.

The sqlservr Command-Line Utility

The sqlservr executable is the program that runs when SQL Server is started. You can use the sqlservr executable to start SQL Server from a command prompt. When you do that, all the startup messages are displayed at the command prompt, and the command prompt session becomes dedicated to the execution of SQL Server.

CAUTION

If you start SQL Server from a command prompt, you cannot stop or pause it by using SSMS, Configuration Manager, or the Services applet in the Control Panel. You should stop the application only from the command prompt window in which SQL Server is running. If you press Ctrl+C, you are asked whether you want to shut down SQL Server. If you close the command prompt window in which SQL Server is running, SQL Server is automatically shut down.

The syntax for the sqlserver utility is as follows:

```
sqlservr [-sinstance_name] [-c] [-dmaster_path] [-f]
     [-eerror_log_path] [-lmaster_log_path] [-m]
     [-n] [-Ttrace#] [-v] [-x] [-gnumber] [-h]
```

Most commonly, you start SQL Server from the command prompt if you need to troubleshoot a configuration problem. The -f option starts SQL Server in minimal configuration mode. This allows you to recover from a change to a configuration setting that prevents SQL Server from starting. You can also use the -m option when you need to start SQL Server in single-user mode, such as when you need to rebuild one of the system databases.

SQL Server functions when started from the command prompt in much the same way as it does when it is started as a service. Users can connect to the server, and you can connect to the server by using SSMS. What is different is that the SQL Server instance running in the command prompt appears as if it is not running in some of the tools. SSMS and SQL Server Service Manager show SQL Server as being stopped because they are polling the SQL Server service, which is stopped when running in the command prompt mode.

TIP

If you simply want to start the SQL Server service from the command prompt, you can use the `NET START` and `NET STOP` commands. These commands are not SQL Server specific but are handy when you want to start or stop SQL Server, especially in a batch file. The SQL Server service name must be referenced after these commands. For example, `NET START MSSQLSERVER` starts the default SQL Server instance.

Removed or Deprecated Utilities in SQL Server 2005

A significant number of command-line utilities have been removed or deprecated in SQL Server 2005. Utilities that have been removed are no longer supported. Those that have been deprecated are still supported but will be removed in a future version of SQL Server. These utilities are not covered in detail in this chapter but are worth noting. You may have used these utilities in prior versions of SQL Server, and you certainly need to know what their status is now. Table 4.3 provides an alphabetic list of the utilities that have been removed or deprecated and provides a brief description of the function of each.

TABLE 4.3 Removed or Deprecated Command-Line Utilities

Utility	Description	Status
`isql`	This utility was used to execute SQL statements, stored procedures, and script files from the command prompt. You should use `sqlcmd` instead.	Removed
`makepipe`	This utility is used to verify a client's connectivity to SQL Server through named pipes.	Deprecated
`odbcping`	This utility is used to test a client machine's ODBC connectivity to the database server.	Deprecated
`osql`	This utility is used to execute SQL statements, stored procedures, and script files from the command prompt. You can use `sqlcmd` instead.	Deprecated
`readpipe`	This utility is used to verify a client's connectivity to SQL Server through named pipes.	Deprecated
`rebuildm`	This utility was used was used to rebuild the `master` database. You can use the `REBUILDDATABASE` option in the `setup.exe` file to achieve the same result.	Removed
`regrebld`	This utility was used to back up and restore the SQL Server Registry entries.	Removed
`sqlmaint`	This utility is used to execute maintenance plans that were created in previous versions of SQL Server.	Deprecated

Summary

SQL Server provides a set of command-line utilities that allow you to execute some of the SQL Server programs from the command prompt. Much of the functionality housed in these utilities is also available in graphical tools, such as SSMS. However, the ability to initiate these programs from the command prompt is invaluable in certain scenarios.

Chapter 5, "SQL Server Profiler," covers a tool that is critical for performance tuning in SQL Server 2005. SQL Server Profiler provides critical insight by monitoring and capturing the activity occurring on a SQL Server instance.

This page intentionally left blank

CHAPTER 5

SQL Server Profiler

IN THIS CHAPTER

This chapter explores the SQL Server Profiler, one of SQL Server's most powerful auditing and analysis tools. The SQL Server Profiler gives you a basic understanding of database access and helps you answer questions such as these:

- Which queries are causing table scans on my invoice history table?
- Am I experiencing deadlocks, and, if so, why?
- What SQL queries is each application submitting?
- Which were the 10 worst-performing queries last week?
- If I implement this alternative indexing scheme, how will it affect my batch operations?

SQL Server Profiler records activity that occurs on a SQL Server instance. The tool has a great deal of flexibility and can be customized for your needs. You can direct SQL Server Profiler to record output to a window, a file, or a table. You can specify which events to trace, the information to include in the trace, how you want it grouped, and what filters you want to apply.

What's New with SQL Server Profiler

There are many new features in and changes to the SQL Server 2005 Profiler. These changes pervade the application and include the following:

- **New Profiler GUI**—The changes to the Profiler GUI application are dramatic. The way that a trace is created, the selection of events, and the general look and feel of the application are quite different than in previous versions.

- **New events**—Many new traceable events have been added to the Profiler. These new events include those that are included in entirely new event categories, such as Broker (Service Broker), CLR, Full Text, and OLE DB. In addition, many new events have been added to categories that existed in past versions, such as `Deadlock graph`, which can be selected from the `Locks` category.
- **New columns**—More than 20 new data columns are available with the SQL Server 2005 Profiler. New columns such as `LineNumber` and `RowCounts` help provide additional insight into what is occurring during a trace.
- **XML enhancement**—Trace, showplan, and deadlock results can now be stored in XML format. Trace results that are saved in XML can be edited and loaded back into the SQL Profiler for replay. Showplan results that are saved in XML can be opened in an SSMS Query Editor window, and the graphical execution plans will be rendered. Deadlock results that are captured in XML can be opened and viewed with a graphical display that helps easily identify the processes involved in the deadlock.
- **Performance Monitor integration**—You can now view trace file output and Performance Monitor files together. The integrated display is synchronized in such a way that you can correlate performance problems with specific activity in your trace output.

SQL Server Profiler Architecture

SQL Server 2005 has both a server and a client-side component for tracing activity on a server. The SQL trace facility is the server-side component that manages queues of events that are initiated by event producers on the server. Extended stored procedures can be used to define the server-side events that are to be captured. These procedures, which define a SQL trace, are discussed later in this chapter, in the section, "Defining Server-Side Traces."

The SQL Profiler is the client-side tracing facility. It comes with a fully functional GUI that allows for real-time auditing of SQL Server events. When it is used to trace server activity, events that are part of a trace definition are gathered at the server. Any filters that are defined as part of the trace definition are applied and the event data is queued for its final destination. The SQL Profiler application is the final destination when client-side tracing is used. The basic elements involved in this process are shown in Figure 5.1.

This figure illustrates the following four steps in the process when tracing from the SQL Server Profiler:

1. Event producers, such as the Query Processor, Lock Manager, ODS, and so on, raise events for the SQL Server Profiler.
2. The filters define the information to submit to SQL Server Profiler. A producer will not send events if the event is not included in the filter.

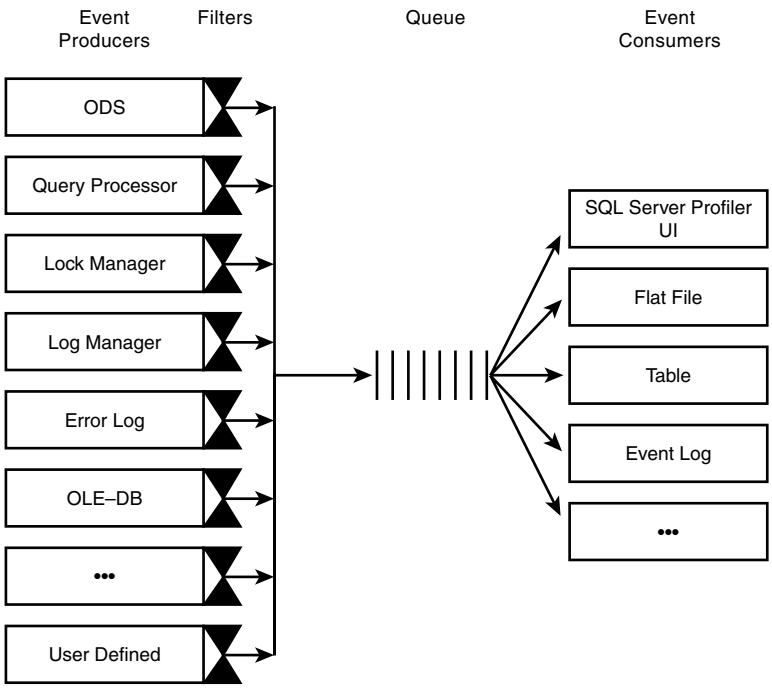

FIGURE 5.1 SQL Server Profiler's architecture.

3. SQL Server Profiler queues all the events.
4. SQL Server Profiler writes the events to each defined consumer, such as a flat file, a table, the Profiler client window, and so on.

In addition to obtaining its trace data from the event producers listed in step 1, you can also configure SQL Profiler so that it obtains its data from a previously saved location. This includes trace data that was saved in a file or table. The "Saving and Exporting Traces" section, later in this chapter, covers using trace files and trace tables in more detail.

Creating Traces

Because SQL Server Profiler can trace numerous events, it is easy to get lost when reading the trace output. You need to roughly determine the information you need and how you want the information grouped. For example, if you want to see the SQL statements that each user is submitting through an application, you could trace incoming SQL statements and group them by user and by application.

Once you have an idea about what you want to trace you should launch the SQL Server Profiler. The Profiler can be launched by selecting Start then SQL Server 2005 then Performance Tools and finally SQL Server Profiler. It can also be launched from within SSMS from the Tools menu. When you launch the Profiler, you are presented with an application window that is basically empty. To start a new trace, you select the File menu and choose New Trace. A connection dialog box is displayed that enables you to enter the connectivity information for the server you want to trace. After the connection is established, the General tab of the Trace Properties window (see Figure 5.2) is displayed.

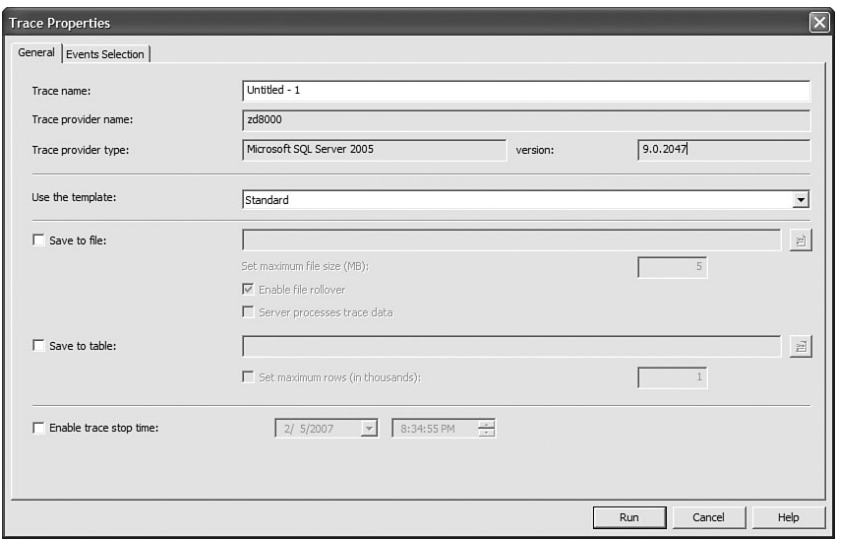

FIGURE 5.2 General trace properties.

The first place you should look when creating a new trace is at the trace templates. These templates contain predefined trace settings that address some common auditing needs. These templates have preset events, data columns, and filters that are targeted at specific profiling scenarios. The available trace templates are found in the template drop-down on the General Properties page and are listed in Table 5.1.

TABLE 5.1 SQL Profiler Templates

Template	Description
`SP_Counts`	Tracks all the stored procedures as they start. No event except for the stored procedure starting is traced.
`Standard`	Traces the completion of SQL statements and remote procedure calls (RPCs) as well as key connection information.
`TSQL`	Traces the start of SQL statements and RPCs. This template is useful for debugging client applications where some of the statements are not completing successfully.

TABLE 5.1 Continued

Template	Description
TSQL_Duration	Traces the total execution time for each completed SQL statement or RPC.
TSQL_Grouped	Traces the start of SQL statements and RPCs, grouped by Application, NTUser, LoginName, and ClientProcessId.
TSQL_Replay	Captures profiling information that is useful for replay. It contains the same type of information as the standard template, but it adds more detail, including cursor and RPC output details.
TSQL_SPs	Traces detailed stored procedures, including the start and completion of each stored procedure. The SQL statements within each procedure are traced as well.
Tuning	A streamlined trace that only tracks the completion of SQL statements and RPCs. The completion events provide duration details that can be useful for performance tuning.

Keep in mind that the templates that come with SQL Server 2005 are not actual traces. They simply provide a foundation for you in creating your own traces. After you select a template, you can modify the trace setting and customize it for your own needs. You can then save the modified template as its own template file that will appear in the template drop-down list for future trace creation.

Trace Name is another property that you can modify on the General tab. Trace Name is a relatively unimportant trace property for future traces. When you create a new trace, you can specify a name for the trace; however, this trace name will not be used again. For instance, if you have a trace definition you like, you can save it as a template file. If you want to run the trace again in the future, you can create a new trace and select the template file that you saved. You will not be selecting the trace to run based on the trace name you entered originally. Trace Name is useful only if you are running multiple traces simultaneously and need to distinguish between them more easily.

TIP

Do yourself a favor and save your favorite trace definitions in your own template. The default set of templates that come with SQL Server are good, but you will most likely want to change the position of a column or add an event that you find yourself using all the time. It is not hard to adjust one of the default templates to your needs each time, but if you save your own template with exactly what you need, it makes it all the more easy. Once you save your own template, you can set it as the default template, and it will be executed by default every time you start the Profiler.

The Save to File and Save to Table options on the General Properties page allow you to define where the trace output is stored. You can save the output to a flat file or a SQL Server table. These options are discussed in more detail later in the chapter, in the section "Saving and Exporting Traces."

The last option on the General Properties screen is the Enable Trace Stop Time option. This scheduling-oriented feature allows you to specify a date and time at which you want to stop tracing. This is handy if you want to start a trace in the evening before you go home. You can set the stop time so that the trace will run for a few hours but won't affect any nightly processing that might occur later in the evening.

Events

The events and data columns that will be captured by your Profiler trace are defined on the Events Selection tab. An example of the Events Selection tab is shown in Figure 5.3.

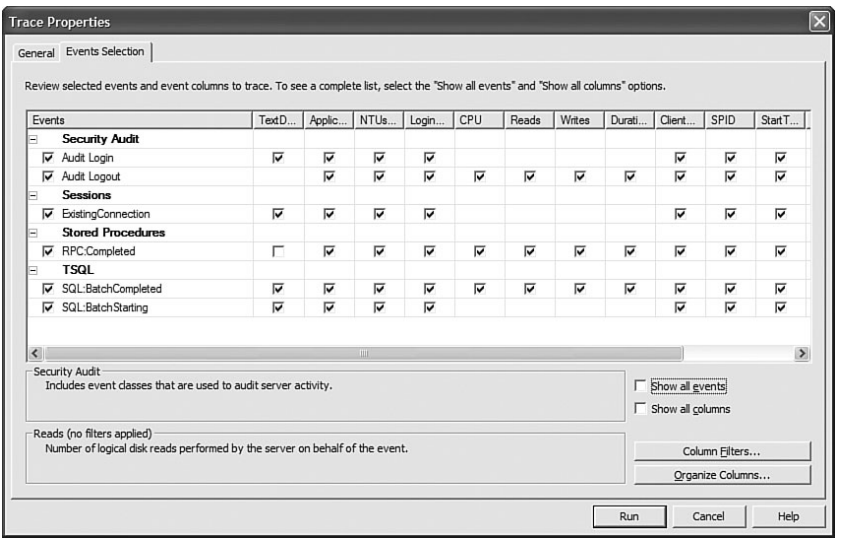

FIGURE 5.3 The Events Selection tab.

The Events Selection tab has changed dramatically in SQL Server 2005. It consolidates the selection of events, data columns, and filters on one tab. In SQL Server 2000, there were three separate tabs for each of these elements. One of the biggest advantages of the SQL Server 2005 Events Selection tab is that you can easily determine which data columns will be populated for each event by looking at the columns that have check boxes available for the event. For example, the Audit Login event has check boxes for `Text Data`, `ApplicationName`, and others but does not have a check box available for `CPU`, `Reads`, `Writes`, and other data columns that are not relevant to the event. For those data columns that have check boxes, you have the option of unchecking the box so that the data column will not be populated for the event when the trace is run.

In SQL Server 2000, if was not obvious what columns would be populated for a particular event. All the data columns were available for selection, regardless of the events chosen. You could determine the columns that were going to be populated for the event by looking at Books Online, but this was more time-consuming and less intuitive than the SQL Server 2005 approach.

On the flip side, you may find that adding events in SQL Server 2005 is less intuitive than it was in SQL Server 2000. When you select a template, the event categories, the selected events in those categories, and the selected columns are displayed in the Events Selection tab. Now, if you want to add additional columns, how do you do it? The answer to this question lies in the Show All Events check box in the lower-right corner of the Events Selection tab. When you click this check box, all the available event categories are listed on the screen. The events and columns that you had previously selected may or may not be visible on the screen. They are not lost, but you may need to scroll down the Events Selection tab to find the event categories that contain the events that you had selected prior to selecting the Show All Events check box.

You will also notice that all the events in the categories in which you had events selected are displayed. In other words, if you had only 2 events selected in the Security Audit category and then selected the Show All Events check box, you see all 42 events listed. The only 2 events that are selected are the ones that you had selected previously, but you need to wade through many events to see them. One upside to this kind of display is that you can easily view all the events for a category and the columns that relate to the events. One possible downside is that the Events Selection tab can be very busy, and it may take a little extra time to find what you are looking for.

TIP

If you capture too many events in one trace, the trace becomes difficult to review. Instead, you can create several traces, one for each type of information that you want to examine, and run them simultaneously. You can also choose to add or remove events after the trace has started. New to SQL Server 2005 is the ability to pause a running trace, change the selected events, and restart the trace without losing the output that was there prior to pausing the trace. In SQL Server 2000, if you didn't want to lose the prior output, you had to save the results to a file or table in this situation.

The selection and viewing of events is made easier by using the tree control that is available on each event. The tree control allows you to expand or compress an event category. When you click the + icon next to a category, all the events are displayed. When you click the - icon, the event category is collapsed to a single row on the display. When an event has been selected for use within a category, the category name is shown in bold. If you want to add all the events in a category to your trace, you can simply right-click the category name and choose the Select Event Category option. You can also remove all events in a category by right-clicking the category name and choosing the Deselect Event Category option.

Understanding what each of the events captures can be a challenging task. You can refer to "SQL Server Event Class Reference" in Books Online for a detailed description, or you can use the simple Help facility available on the Events Selection tab. The Events Selection tab has a Help facility that describes each of the events and categories. The Help text is displayed on the Events Selection tab below the list of available events. When you

mouse over a particular event or event category, a description of that item is shown. This puts the information you need at your fingertips.

NOTE

If you are going to use SQL Server Profiler, you should spend some time getting to know the events first and the type of output that Profiler generates. You should do this first in a development environment or standalone environment where the Profiler's effect on performance does not matter. It's a good idea to start a trace with a few events at a time and execute some relevant statements to see what is displayed for each event. You will soon realize the strength of the SQL Server Profiler and the type of valuable information it can return.

Data Columns

The columns of information that are captured in a Profiler trace are determined by the Data Columns selected. The Events Selection tab has the functionality you need to add columns, organize the columns, and apply filters on the data that is returned in these columns. As mentioned earlier, you can select and deselect the available columns for a particular event by using the check boxes displayed for the listed events. To understand what kind of information a column is going to return, you can simply mouse over the column, and Help for that item is displayed in the second Help box below the event list. Figure 5.4 shows an example of the Help output. In this particular case, the mouse pointer is over the `ApplicationName` column that is returned for the `SQL:BatchCompleted` event. The first Help box displays information about the `SQL:BatchCompleted` event. The second Help box shows information about the data column.

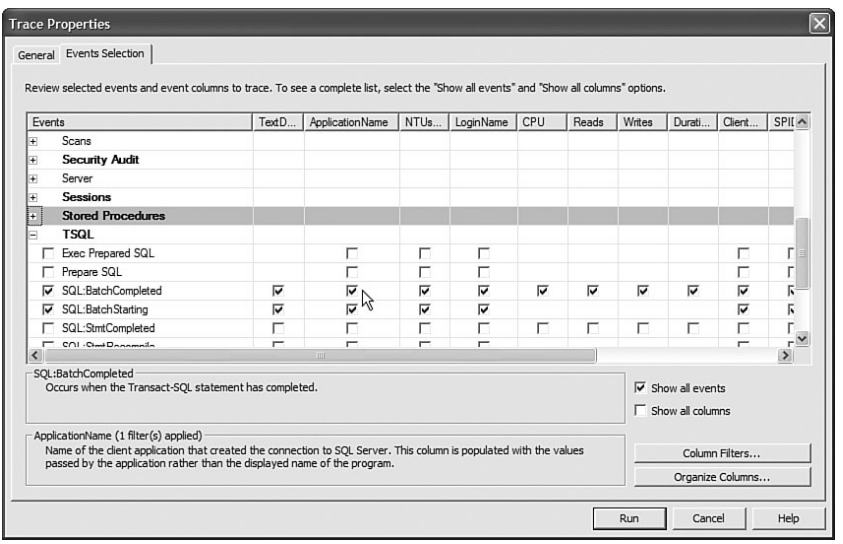

FIGURE 5.4 Help for data columns on the Events Selection tab.

Keep in mind that there is a default set of columns that is displayed for each event. You can view additional columns by selecting the Show All Columns check box. When you choose this option, an additional set of columns is displayed in the Events Selection tab. The additional columns are shown with a dark gray background, and you may need to scroll to the right on the Events Selection tab in order to see them. Figure 5.5 shows an example of the additional columns that are displayed for the `Cursors` event when the Show All Columns option is used. Some of the additional columns that are available for selection in this example are `DatabaseID` and `DatabasedName`.

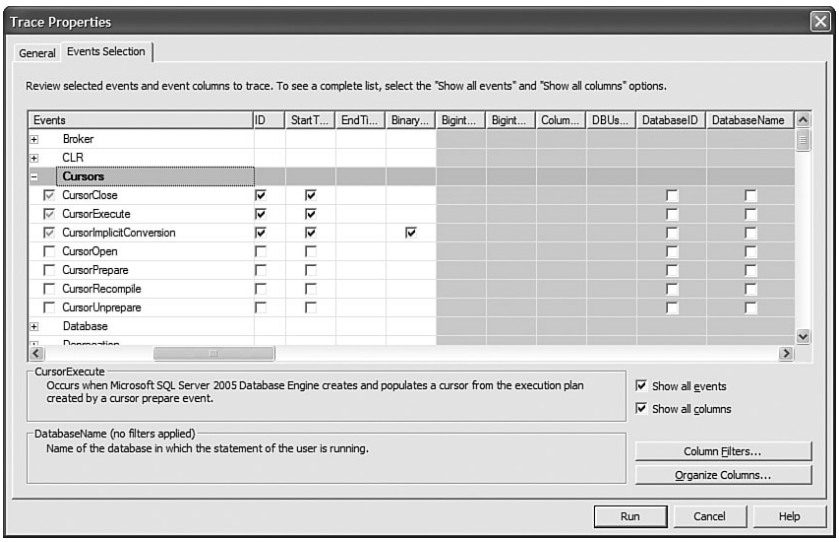

FIGURE 5.5 Additional columns displayed with the Show All Columns option.

To organize the columns that you have selected, you can choose the Organize Columns selection on the Events Selection tab. This Organize Columns window allows you to change the order of the columns in the trace output as well as group the data by selected columns. Figure 5.6 shows an example of the Organize Columns window with the groups and columns that are selected by default when you use the `TSQL_Grouped` template.

To change the order of a column, you simply select the column in the list and use the Up or Down buttons to move it. The same movement can be done with columns that have been selected for grouping. You add columns to groups by selecting the column in the data list and clicking the Up button until the column is moved out of the Columns list and into the Groups list. For example, in Figure 5.6, you can group the `SPID` column by selecting it and clicking the Up button one time, which places it in the Groups tree structure instead of the Columns tree structure.

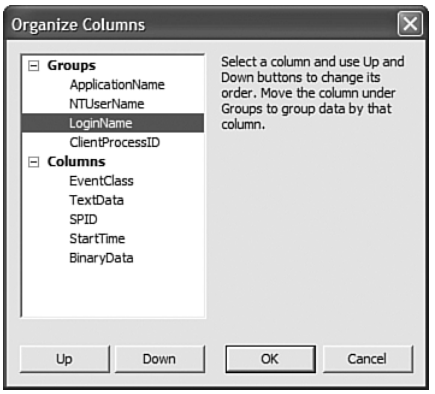

FIGURE 5.6 Organizing columns in the Events Selection tab.

TIP

You can select a particular column for all events by right-clicking the column header in the Events Selection tab and choosing the Select Column option. This causes all the check boxes on the grid to be selected. To remove a column from all events, you right-click the column header and choose Deselect Column.

The number of columns selected for grouping and the order of the columns are both important factors in the way the trace data will be displayed. If you choose only one column for grouping, the trace window displays events grouped by the values in the grouped data column and collapses all events under it. For example, if you group by `DatabaseId`, the output in the trace window grid displays `DatabaseId` as the first column, with a + sign next to each `DatabaseId` that has received events. The number displayed to the right of the event in parentheses shows the number of collapsed events that can be viewed by clicking on the + sign. Figure 5.7 shows an example of the trace output window that has been grouped by `DatabaseId` only. The database with a `DatabaseId` equal to `6` is shown at the bottom of the grid in this example. The grid has been expanded and some of the 20 events that were captured for this `DatabaseId` are shown.

If you select multiple columns for grouping, the output in the trace window is ordered based on the columns in the grouping. The events are not rolled up like a single column, but the trace output grid automatically places the incoming events in the proper order in the output display.

TIP

The organization of columns in a trace can happen after a trace has been defined and executed. If you save the trace to a file or a table, you can open it later and specify whatever ordering or grouping you want to reorganize the output. This flexibility gives you almost endless possibilities for analyzing the trace data.

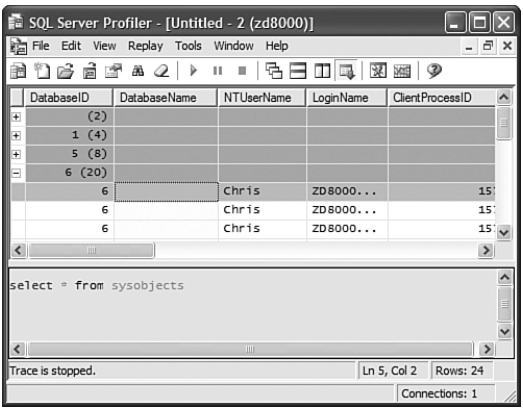

FIGURE 5.7 Grouping on a single column.

Filters

Filters restrict the event data that is returned in your trace output. You can filter the events that are captured by the SQL Profiler via the Column Filters button on the Events Selection tab. The SQL Server 2005 Edit Filter window that is displayed is much different than that which was used in SQL Server 2000. An example of the new window is shown in Figure 5.8. All the available columns for the trace are shown on the right side of the Edit Filter window, along with any defined filters. Those columns that have filters on them have a filter icon displayed next to the column in the column list.

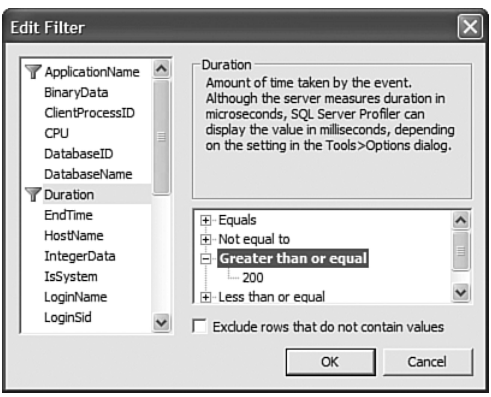

FIGURE 5.8 Editing filter properties.

The filtering options in SQL Server 2005 are similar to those available in SQL Server 2000. Which options are available depends on the type of column you are filtering on. The following list describes the different filtering options:

- **Like/Not Like**—This gives you the ability to include or exclude events based on a wildcard. You should use the % character as your wildcard character and press Enter to create an entry space for another value. For example, with the `ApplicationName` filter, you can specify `Like Microsoft%`, and you will get only those events related to applications that match the wildcard, such as Microsoft SQL Server Management Studio. This filtering option is available for text data columns and data columns that contain name information, such as `NTUserName` and `ApplicationName`.
- **Equals/Not Equal To/Greater Than or Equal/Less Than or Equal**—Filters with this option have all four of these conditions available. For the Equals and Not Equal To conditions, you can specify a single value or a series of values. For the other conditional types, a single value is supplied. For example, you can filter on `DataBaseID` and input numeric values under the `Equals To` node of the filtering tree. This filtering option is available for numeric data columns such as `Duration`, `IndexId`, and `ObjectId`.
- **Greater Than/Less Than**—This type of filtering option is available only on time-based data columns. This includes `StartTime` and `EndTime` filters. These filters expect date formats of the form YYYY-MM-DD or YYYY-MM-DD HH:MM:SS.

Each data column can use one of these three filtering options. When you click the data column that is available for filtering, you see the filtering options for that column displayed in the right pane of the Edit Filter window. You enter the values on which you want to filter in the data entry area on the filter tree. This input area is shown when you select a specific filtering option. For multiple filter values, you press the Enter key after you enter each value. This causes a new data entry area to appear below the value you were on.

CAUTION

Filters that are applied to columns that are not available or selected for an event do not prevent the event data from being returned. For example, if you place a filter on the `ObjectName` column and choose the `SQL:StmtStarting` event as part of your trace, the event data is not filtered because `ObjectName` is not a valid column for that event. This may seem relatively intuitive, but it is something to consider when you are receiving output from a trace that you believe should have been filtered out.

Also, be careful when specifying multiple filter values and consider the Boolean logic that is applied to them. When specifying multiple values for the `Like` filter, the values are evaluated with an `OR` condition. For example, if you create a filter on `ObjectName` and have a `Like` filter with values of `A%`, `B%`, and `C%`, the filter will return object names that start with A or B or C. When you use the `Not Like` filter, the `AND` condition is used on multiple values. For example, `Not Like` filter values for `ObjectName` of `A%` and `C%` results in objects with names that do not start with A and object names that do not start with C.

Executing Traces and Working with Trace Output

After you have defined the events and columns that you want to capture in a trace, you can execute the Profiler trace. To do so, you click the Run button on the Trace Properties window, and the Profiler GUI starts capturing the events you have selected. The GUI contains a grid that is centrally located on the Profiler window, and newly captured events are scrolled on the screen as they are received. Figure 5.9 shows a simple example of the Profiler screen with output from an actively running trace.

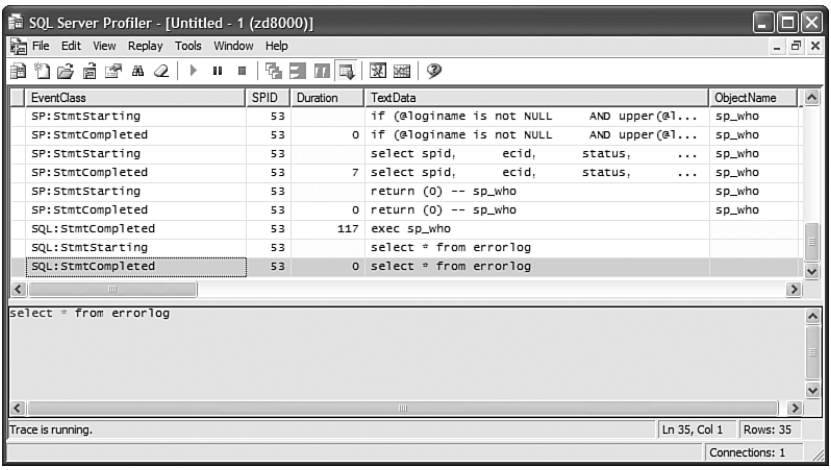

FIGURE 5.9 The Profiler GUI with an active trace.

The Profiler GUI provides many different options for dealing with an actively running trace. You can turn off scrolling on the trace, pause the trace, stop the trace, and view the properties of an actively running trace. You can find strings within the trace output, and you can even move the columns around in the display so that they are displayed in a different order. These options provide a great deal of flexibility and allow you to focus on the output that is most important to you.

Saving and Exporting Traces

In many cases, you will want to save or export the trace output generated by a Profiler trace. The output can be analyzed, replayed, imported, or manipulated at a later time after it has been saved. Trace output can be saved as the trace is running or saved after it has been generated to the Profiler GUI. The Trace Properties window provides options for saving trace output while the trace is running. The options are defined using the Save to File and Save to Table options on the General tab of the Trace Properties dialog. You can save to a file, a table, or both a table and a file. Figure 5.10 shows an example of a trace that will save to both a file and a table while it is executing.

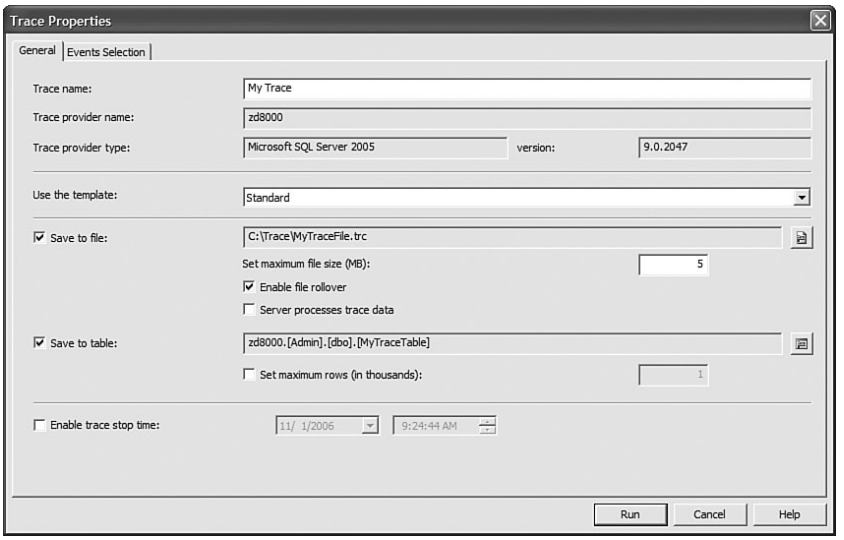

FIGURE 5.10 Saving trace output while a trace is running.

Saving Trace Output to a File

When you save a running trace to a file, you have several options for controlling the output. One option you should always consider is the Set Maximum File Size (MB) option. This option prevents a trace output file from exceeding the specified size. This helps make the file more manageable and, more importantly, it can save you from having a trace file gobble up all the disk space on the drive you are writing to. Remember that the amount of trace data that is written to a file on a busy production system can be extensive. You can also use this file size option in conjunction with the Enable File Rollover option. When the Enable File Rollover option is used, the trace does not stop when the file size maximum is met. Instead, a new trace file is created, and the output is generated to that file until it reaches the file size maximum.

Saving Trace Output to a Table

The Save to Table option writes the trace output directly to a SQL Server table as the trace is running. Having the data in a SQL table provides a great deal of flexibility for analyzing the data. You can use the full power of Transact-SQL against the table, including sorting, grouping, and more complex search conditions than are not available through the SQL Profiler filters.

You need to consider both the disk space requirements and the impact on performance when the Save to Table option is used. The Profiler provides an option, Set Maximum Rows (in Thousands), to limit the amount of output generated from the trace. The performance impact depends on the volume of data that is being written to the table. Generally, writing the trace output to a table should be avoided with high-volume SQL

servers. The best option for high-volume servers is to first write the trace output to a file and then import the file to a trace table at a later time.

Saving the Profiler GUI Output

Another option for saving trace output occurs after trace output has been generated to the Profiler GUI and the trace has been stopped. Similar to the save options for an executing trace, the GUI output can be saved to a file or table. You access the options to save the GUI output by selecting File, Save As. The Trace File and Trace Table options are used to save to a file or table consecutively. With SQL Server 2005, you can also save the output to an XML file. The Trace XML File and Trace XML File for Replay options generate XML output that can be edited or used as input for replay with the SQL Server Profiler.

NOTE

Two distinct save operations are available in the SQL Profiler. You can save trace events to a file or table as just described, or you can save a trace definition in a template file. The Save As Trace Table and Save As Trace File options are for saving trace events to a file. The Save As Trace Template option saves the trace definition. Saving a trace template saves you the trouble of having to go through all the properties each time to set up the events, data columns, and filters for your favorite traces.

An alternative to saving all the event data associated with a particular trace is to select specific event rows from the SQL Profiler windows. You can capture all the trace information associated with a trace row by selecting a row in the trace output window of Profiler and choosing Edit, Copy. Or, you can just copy the event text (typically a SQL statement) by selecting the row, highlighting the text in the lower pane, and using the Copy option. You can then paste this data into SSMS or the tool of your choice for further execution and more detailed analysis. This can be particularly useful during performance tuning. After you identify the long-running statement or procedure, you can copy the SQL, paste it into SSMS, and display the query plan to determine why the query was running so long.

Importing Trace Files

A trace saved to a file or table can be read back into SQL Profiler at a later time for more detailed analysis or to replay the trace on the same SQL Server or another SQL Server instance. You can import data from a trace file or trace table by choosing File, Open and then selecting either a trace file or trace table. If you choose to open a trace file, you are presented with a dialog box to locate the trace file on the local machine. If you choose to import a trace table, you are first presented with a connection dialog box to specify the SQL Server name, the login ID, and the password to connect to it. When you are successfully connected, you are presented with a dialog box to specify the database and the name of the trace table you want to import from. After you specify the trace file or trace table to import into Profiler, the entire contents of the file or table are read in and displayed in a Profiler window.

You may find that large trace files or trace tables are difficult to analyze, and you may just want to analyze events associated with a specific application or table, or a specific types of queries. To limit the amount of information displayed in the Profiler window, you can filter out the data displayed via the Properties dialog. You can choose which events and data columns you want to display and also specify conditions in the Filters tab to limit the rows displayed from the trace file or trace table. These options do not affect the information stored in the trace file or trace table—only what information is displayed in the Profiler window.

Importing a Trace File into a Trace Table

Although you can load a trace file directly into Profiler for analysis, very large files can be difficult to analyze. Profiler loads an entire file. For large files, this can take quite a while, and the responsiveness of Profiler might not be the best. Multiple trace output files for a given trace can also be cumbersome and difficult to manage when those files are large.

You can use the trace filters to limit which rows are displayed but not which rows are imported into Profiler. You often end up with a bunch of rows displayed with no data in the columns you want to analyze. In addition, while the filters allow you to limit which rows are displayed, they don't really provide a means of running more complex reports on the data, such as generating counts of events or displaying the average query duration.

Fortunately, SQL Server 2005 provides a way for you to selectively import a trace file into a trace table. When importing a trace file into a trace table, you can filter the data before it goes into the table as well as combine multiple files into a single trace table. Once the data is in a trace table, you can load the trace table into Profiler or write your own queries and reports against the trace table for more detailed analysis than is possible in Profiler.

Microsoft SQL Server includes some built-in user-defined functions for working with Profiler traces. The `fn_trace_gettable` function is used to import trace file data into a trace table. The following is the syntax for this function:

```
fn_trace_gettable( [ @filename = ] filename , [ @numfiles = ] number_files )
```

This function returns the contents of the specified file as a table result set. You can use the result set from this function just as you would any table. By default, the function returns all possible Profiler columns, even if no data was captured for the column in the trace. To limit the columns returned, you specify the list of columns in the query. If you want to limit the rows retrieved from the trace file, you specify your search conditions in the `WHERE` clause. If your Profiler trace used rollover files to split the trace across multiple files, you can specify the number of files you want it to read in. If the default value of `default` is used, all rollover files for the trace are loaded. Listing 5.1 provides an example of creating and populating a trace table from a trace file, using `SELECT INTO`, and then adding rows by using an `INSERT` statement. Note that this example limits the columns and rows returned by specifying a column list and search conditions in the `WHERE` clause.

LISTING 5.1 Creating and Inserting Trace Data into a Trace Table from a Trace File

```
/*****************************************************************
** NOTE - you will need to edit the path/filename on your system if
**        you use this code to load your own trace files
*****************************************************************/

select EventClass,
       EventSubClass,
       TextData = convert(varchar(8000), TextData),
       BinaryData,
       ApplicationName,
       Duration,
       StartTime,
       EndTime,
       Reads,
       Writes,
       CPU,
       ObjectID,
       IndexID,
       NestLevel
   into TraceTable
   FROM ::fn_trace_gettable('c:\temp\sampletrace_20060826_0232.trc', default)
   where TextData is not null
      or EventClass in (16, --  Attention
                        25, -- Lock:Deadlock
                        27, -- Lock:Timeout
                        33, -- Exception
                        58, -- Auto Update Stats
                        59, -- Lock:Deadlock Chain
                        79, -- Missing Column Statistics
                        80, -- Missing Join Predicate
                        92, -- Data File Auto Grow
                        93, -- Log File Auto Grow
                        94, -- Data File Auto Shrink
                        95) -- Log File Auto Shrink

Insert into TraceTable (EventClass, EventSubClass,
            TextData, BinaryData,
            ApplicationName, Duration, StartTime, EndTime, Reads, Writes,
            CPU, ObjectID, IndexID, nestlevel)
     select EventClass, EventSubClass,
            TextData = convert(varchar(7900), TextData), BinaryData,
            ApplicationName, Duration, StartTime, EndTime, Reads, Writes,
            CPU, ObjectID, IndexID, nestlevel
        FROM ::fn_trace_gettable('c:\temp\sampletrace_20060826_0108.trc', -1)
```

LISTING 5.1 Continued

```
        where TextData is not null
           or EventClass in (16, --  Attention
                             25, -- Lock:Deadlock
                             27, -- Lock:Timeout
                             33, -- Exception
                             58, -- Auto Update Stats
                             59, -- Lock:Deadlock Chain
                             79, -- Missing Column Statistics
                             80, -- Missing Join Predicate
                             92, -- Data File Auto Grow
                             93, -- Log File Auto Grow
                             94, -- Data File Auto Shrink
                             95) -- Log File Auto Shrink
go
```

Once the trace file is imported into a trace table, you can open the trace table in Profiler or run your own queries against the trace table from a Query Editor window in SSMS. For example, the following query returns the number of lock timeouts encountered for each table during the period the trace was running:

```
select object_name(ObjectId), count(*)
    from TraceTable
    where EventClass = 27 -- Lock:Timout Event
    group by object_name(ObjectId)
go
```

Analyzing Trace Output with the Database Engine Tuning Advisor

In addition to being able to manually analyze traces in Profiler, you can also use the new Database Engine Tuning Advisor to analyze the queries captured in a trace and recommend changes to your indexing scheme. The Database Engine Tuning Advisor is a replacement for the Index Tuning Wizard. You can invoke it from the Tools menu in SQL Profiler. The Database Engine Tuning Advisor can read in a trace that was previously saved to a table or a file. This allows you to capture a workload, tune the indexing scheme, and rerun the trace to determine whether the index changes improved performance as expected.

Because the Database Engine Tuning Advisor analyzes SQL statements, you need to make sure that the trace includes one or more of the following events:

```
SP:StmtCompleted
SP:StmtStarting
SQL:BatchCompleted
SQL:BatchStarting
```

```
SQL:StmtCompleted
SQL:StmtStarting
```

One of each class (one `SP:` and one `SQL:`) is sufficient to capture dynamic SQL statements and statements embedded in stored procedures. You should also make sure that the trace includes the text data column, which contains the actual queries.

The Database Engine Tuning Advisor analyzes the trace and gives you recommendations, along with an estimated improvement-in-execution time. You can choose to create indexes now or at a later time, or you can save the `CREATE INDEX` commands to a script file.

Replaying Trace Data

To replay a trace, you must have a trace saved to a file or a table. The trace must be captured with certain trace events to enable playback. The required events are captured by default if you use the Profiler template `TSQL_Replay`. You can define a trace to be saved when you create or modify the trace definition. You can also save the current contents of the trace window to a file or table by using the Save As Trace File or Save As Trace Table options in the File menu.

To replay a saved trace, you use the File, Open menu to open a trace file or a trace table. After you select the type of trace to replay, a grid with the trace columns selected in the original trace is displayed. At this point, you can either start the replay of the trace step-by-step or complete execution of the entire trace. The options for replaying the trace are found under the Replay menu. When you start the replay of the trace, the Connect to Server dialog is displayed, enabling you to choose the server that you want to replay the traces against. Once you are connected to a server, a Replay Configuration dialog box like the one shown in Figure 5.11 is displayed.

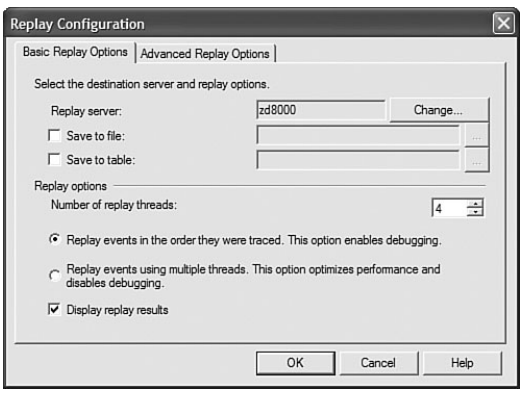

FIGURE 5.11 Basic replay options.

The first replay option, which is enabled by default, replays the trace in the same order in which it was captured and allows for debugging. The second option takes advantage of multiple threads; it optimizes performance but disables debugging. A third option involves specifying whether to display the replay results. You would normally want to see the results, but for large trace executions, you might want to forgo displaying the results and send them to an output file instead.

If you choose the option that allows for debugging, you can execute the trace in a manner similar to many programming tools. You can set breakpoints, step through statements one at a time, or position the cursor on a statement within the trace and execute the statements from the beginning of the trace to the cursor position.

NOTE

Automating testing scripts is another important use of the SQL Profiler Save and Replay options. For instance, a trace of a heavy production load can be saved and rerun against a new release of the database to ensure that the new release has similar or improved performance characteristics and returns the same data results. The saved traces can help make regression testing much easier.

You also have the option of specifying advanced replay options in SQL Server 2005. These are new options that are found on the Advanced Replay Options tab of the Replay Configuration dialog box (see Figure 5.12).

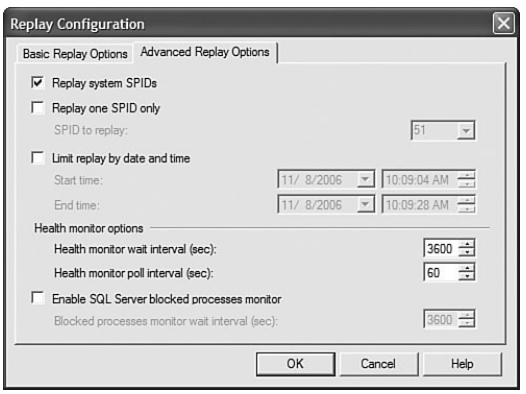

FIGURE 5.12 Advanced replay options.

The first two options on the Advanced Replay Options tab relate to the system process IDs (SPIDs) that will be targeted for replay. If the Replay System SPIDs option is selected, then the trace events for every SPID in the trace file will be replayed. If you want to target activity for a specific SPID, you should choose the Replay One SPID Only option and select the SPID from the drop-down menu. You can also limit the events that will be replayed based on the timing of the events. If you want to replay a specific time-based

section of the trace, you can use the Limit Replay by Date and Time option. Only those trace events that fall between the data range you specify will be replayed.

The last set of advanced options are geared toward maintaining the health of the server on which you are replaying the trace. The Health Monitor Wait Interval (sec) option determines the amount of time a thread can run during replay before being terminated. This helps avoid an excessive drain on the server's resources. The Health Monitor Poll Interval (sec) option determines how often the health monitor will poll for threads that should be terminated. The last advanced option on the screen relates to blocked processes. When it is enabled, the monitor will poll for blocked processes according to the interval specified.

Defining Server-Side Traces

Much of the SQL Server Profiler functionality can also be initiated through a set of system stored procedures. Through these procedures, you can define a server-side trace that can be run automatically or on a scheduled basis, such as via a scheduled job, instead of through the Profiler GUI. Server-side traces are also useful if you are tracing information over an extended period of time or are planning on capturing a large amount of trace information. The overhead of running a server-side trace is less than that of running a client-side trace with Profiler.

To start a server-side trace, you need to define the trace by using the trace-related system procedures. These procedures can be called from within a SQL Server stored procedure or batch. You define a server-side trace by using the following four procedures:

- **`sp_trace_create`**—This procedure is used to create the trace definition. It sets up the trace and defines the file to store the captured events. `sp trace create` returns a trace ID number that you'll need to reference from the other three procedures to further define and manage the trace.
- **`sp_trace_setevent`**—You need to call this procedure once for each data column of every event that you want to capture.
- **`sp_trace_setfilter`**—You call this procedure once for each filter you want to define on an event data column.
- **`sp_trace_setstatus`**—After the trace is defined, you call this procedure to start, stop, or remove the trace. You must stop and remove a trace definition before you can open and view the trace file.

You will find that manually creating procedure scripts for tracing can be rather tedious. Much of the tedium is due to the fact that many numeric parameters drive the trace execution. For example, the `sp_trace_setevent` procedure accepts an `eventid` and a `columnid` that determine what event data will be captured. Fortunately, SQL Server 2005 provides a set of catalog views that contain these numeric values and what they represent. The `sys.trace_categories` catalog view contains the event categories. The `sys.trace_events` catalog view contains the trace events, and `sys.trace_columns` contains the trace

columns. The following SELECT statement utilizes two of these system views to return the available events and their related categories:

```
select e.trace_event_id, e.name 'Event Name', c.name 'Category Name'
 from sys.trace_events e
  join sys.trace_categories c on e.category_id = c.category_id
 order by e.trace_event_id
```

The results of this SELECT statement are shown in Table 5.2.

TABLE 5.2 Trace Events and Their Related Categories

trace_event_id	Event Name	Category Name
10	RPC:Completed	Stored Procedures
11	RPC:Starting	Stored Procedures
12	SQL:BatchCompleted	TSQL
13	SQL:BatchStarting	TSQL
14	Audit Login	Security Audit
15	Audit Logout	Security Audit
16	Attention	Errors and Warnings
17	ExistingConnection	Sessions
18	Audit Server Starts And Stops	Security Audit
19	DTCTransaction	Transactions
20	Audit Login Failed	Security Audit
21	EventLog	Errors and Warnings
22	ErrorLog	Errors and Warnings
23	Lock:Released	Locks
24	Lock:Acquired	Locks
25	Lock:Deadlock	Locks
26	Lock:Cancel	Locks
27	Lock:Timeout	Locks
28	Degree of Parallelism (7.0 Insert)	Performance
33	Exception	Errors and Warnings
34	SP:CacheMiss	Stored Procedures
35	SP:CacheInsert	Stored Procedures
36	SP:CacheRemove	Stored Procedures
37	SP:Recompile	Stored Procedures
38	SP:CacheHit	Stored Procedures
39	Deprecated	Stored Procedures
40	SQL:StmtStarting	TSQL
41	SQL:StmtCompleted	TSQL
42	SP:Starting	Stored Procedures
43	SP:Completed	Stored Procedures
44	SP:StmtStarting	Stored Procedures

TABLE 5.2 Continued

trace_event_id	Event Name	Category Name
45	SP:StmtCompleted	Stored Procedures
46	Object:Created	Objects
47	Object:Deleted	Objects
50	SQLTransaction	Transactions
51	Scan:Started	Scans
52	Scan:Stopped	Scans
53	CursorOpen	Cursors
54	TransactionLog	Transactions
55	Hash Warning	Errors and Warnings
58	Auto Stats	Performance
59	Lock:Deadlock Chain	Locks
60	Lock:Escalation	Locks
61	OLEDB Errors	OLEDB
67	Execution Warnings	Errors and Warnings
68	Showplan Text (Unencoded)	Performance
69	Sort Warnings	Errors and Warnings
70	CursorPrepare	Cursors
71	Prepare SQL	TSQL
72	Exec Prepared SQL	TSQL
73	Unprepare SQL	TSQL
74	CursorExecute	Cursors
75	CursorRecompile	Cursors
76	CursorImplicitConversion	Cursors
77	CursorUnprepare	Cursors
78	CursorClose	Cursors
79	Missing Column Statistics	Errors and Warnings
80	Missing Join Predicate	Errors and Warnings
81	Server Memory Change	Server
82	UserConfigurable:0	User configurable
83	UserConfigurable:1	User configurable
84	UserConfigurable:2	User configurable
85	UserConfigurable:3	User configurable
86	UserConfigurable:4	User configurable
87	UserConfigurable:5	User configurable
88	UserConfigurable:6	User configurable
89	UserConfigurable:7	User configurable
90	UserConfigurable:8	User configurable
91	UserConfigurable:9	User configurable
92	Data File Auto Grow	Database

TABLE 5.2 Continued

trace_event_id	Event Name	Category Name
93	Log File Auto Grow	Database
94	Data File Auto Shrink	Database
95	Log File Auto Shrink	Database
96	Showplan Text	Performance
97	Showplan All	Performance
98	Showplan Statistics Profile	Performance
100	RPC Output Parameter	Stored Procedures
102	Audit Database Scope GDR Event	Security Audit
103	Audit Schema Object GDR Event	Security Audit
104	Audit Addlogin Event	Security Audit
105	Audit Login GDR Event	Security Audit
106	Audit Login Change Property Event	Security Audit
107	Audit Login Change Password Event	Security Audit
108	Audit Add Login to Server Role Event	Security Audit
109	Audit Add DB User Event	Security Audit
110	Audit Add Member to DB Role Event	Security Audit
111	Audit Add Role Event	Security Audit
112	Audit App Role Change Password Event	Security Audit
113	Audit Statement Permission Event	Security Audit
114	Audit Schema Object Access Event	Security Audit
115	Audit Backup/Restore Event	Security Audit
116	Audit DBCC Event	Security Audit
117	Audit Change Audit Event	Security Audit
118	Audit Object Derived Permission Event	Security Audit
119	OLEDB Call Event	OLEDB
120	OLEDB QueryInterface Event	OLEDB
121	OLEDB DataRead Event	OLEDB
122	Showplan XML	Performance
123	SQL:FullTextQuery	Performance
124	Broker:Conversation	Broker
125	Deprecation Announcement	Deprecation
126	Deprecation Final Support	Deprecation
127	Exchange Spill Event	Errors and Warnings
128	Audit Database Management Event	Security Audit
129	Audit Database Object Management Event	Security Audit
130	Audit Database Principal Management Event	Security Audit

TABLE 5.2 Continued

trace_event_id	Event Name	Category Name
131	Audit Schema Object Management Event	Security Audit
132	Audit Server Principal Impersonation Event	Security Audit
133	Audit Database Principal Impersonation Event	Security Audit
134	Audit Server Object Take Ownership Event	Security Audit
135	Audit Database Object Take Ownership Event	Security Audit
136	Broker:Conversation Group	Broker
137	Blocked process report	Errors and Warnings
138	Broker:Connection	Broker
139	Broker:Forwarded Message Sent	Broker
140	Broker:Forwarded Message Dropped	Broker
141	Broker:Message Classify	Broker
142	Broker:Transmission	Broker
143	Broker:Queue Disabled	Broker
144	Broker:Mirrored Route State Changed	Broker
146	Showplan XML Statistics Profile	Performance
148	Deadlock graph	Locks
149	Broker:Remote Message Acknowledgement	Broker
150	Trace File Close	Server
152	Audit Change Database Owner	Security Audit
153	Audit Schema Object Take Ownership Event	Security Audit
155	FT:Crawl Started	Full text
156	FT:Crawl Stopped	Full text
157	FT:Crawl Aborted	Full text
158	Audit Broker Conversation	Security Audit
159	Audit Broker Login	Security Audit
160	Broker:Message Undeliverable	Broker
161	Broker:Corrupted Message	Broker
162	User Error Message	Errors and Warnings
163	Broker:Activation	Broker
164	Object:Altered	Objects
165	Performance statistics	Performance
166	SQL:StmtRecompile	TSQL
167	Database Mirroring State Change	Database
168	Showplan XML For Query Compile	Performance

TABLE 5.2 Continued

trace_event_id	Event Name	Category Name
169	Showplan All For Query Compile	Performance
170	Audit Server Scope GDR Event	Security Audit
171	Audit Server Object GDR Event	Security Audit
172	Audit Database Object GDR Event	Security Audit
173	Audit Server Operation Event	Security Audit
175	Audit Server Alter Trace Event	Security Audit
176	Audit Server Object Management Event	Security Audit
177	Audit Server Principal Management Event	Security Audit
178	Audit Database Operation Event	Security Audit
180	Audit Database Object Access Event	Security Audit
181	TM: Begin Tran starting	Transactions
182	TM: Begin Tran completed	Transactions
183	TM: Promote Tran starting	Transactions
184	TM: Promote Tran completed	Transactions
185	TM: Commit Tran starting	Transactions
186	TM: Commit Tran completed	Transactions
187	TM: Rollback Tran starting	Transactions
188	TM: Rollback Tran completed	Transactions
189	Lock:Timeout (timeout > 0)	Locks
190	Progress Report: Online Index Operation	Progress Report
191	TM: Save Tran starting	Transactions
192	TM: Save Tran completed	Transactions
193	Background Job Error	Errors and Warnings
194	OLEDB Provider Information	OLEDB
195	Mount Tape	Server
196	Assembly Load	CLR
198	XQuery Static Type	TSQL
199	QN: Subscription	Query Notifications
200	QN: Parameter table	Query Notifications
201	QN: Template	Query Notifications
202	QN: Dynamics	Query Notifications

The numeric IDs for the trace columns can be obtained from the `sys.trace_columns` catalog view, as shown in the following example:

```
select trace_column_id, name 'Column Name', type_name 'Data Type'
 from sys.trace_columns
 order by trace_column_id
```

Table 5.3 shows the results of this SELECT statement and lists all the available trace columns.

TABLE 5.3 Trace Columns Available for a Server-Side Trace

trace_column_id	Column Name	Data Type
1	TextData	text
2	BinaryData	image
3	DatabaseID	int
4	TransactionID	bigint
5	LineNumber	int
6	NTUserName	nvarchar
7	NTDomainName	nvarchar
8	HostName	nvarchar
9	ClientProcessID	int
10	ApplicationName	nvarchar
11	LoginName	nvarchar
12	SPID	int
13	Duration	bigint
14	StartTime	datetime
15	EndTime	datetime
16	Reads	bigint
17	Writes	bigint
18	CPU	int
19	Permissions	bigint
20	Severity	int
21	EventSubClass	int
22	ObjectID	int
23	Success	int
24	IndexID	int
25	IntegerData	int
26	ServerName	nvarchar
27	EventClass	int
28	ObjectType	int
29	NestLevel	int
30	State	int
31	Error	int
32	Mode	int
33	Handle	int
34	ObjectName	nvarchar
35	DatabaseName	nvarchar
36	FileName	nvarchar
37	OwnerName	nvarchar

TABLE 5.3 Continued

trace_column_id	Column Name	Data Type
38	RoleName	nvarchar
39	TargetUserName	nvarchar
40	DBUserName	nvarchar
41	LoginSid	image
42	TargetLoginName	nvarchar
43	TargetLoginSid	image
44	ColumnPermissions	int
45	LinkedServerName	nvarchar
46	ProviderName	nvarchar
47	MethodName	nvarchar
48	RowCounts	bigint
49	RequestID	int
50	XactSequence	bigint
51	EventSequence	int
52	BigintData1	bigint
53	BigintData2	bigint
54	GUID	uniqueidentifier
55	IntegerData2	int
56	ObjectID2	bigint
57	Type	int
58	OwnerID	int
59	ParentName	nvarchar
60	IsSystem	int
61	Offset	int
62	SourceDatabaseID	int
63	SqlHandle	image
64	SessionLoginName	nvarchar
65	PlanHandle	image

You have to call the `sp_trace_setevent` procedure once for each data column you want captured for each event in the trace. Based on the number of events and number of columns, you can see that this can result in a lot of executions of the `sp_trace_setevent` procedure for a larger trace.

To set up filters, you must pass the column ID, the filter value, and numeric values for the logical operator and the column operator to the `sp_trace_setfilter` procedure. The logical operator can be either `0` or `1`. A value of `0` indicates that the specified filter on the column should be `AND`ed with any other filters on the column, while a value of `1` indicates that the `OR` operator should be applied. Table 5.4 describes the values allowed for the column operators.

TABLE 5.4 Column Operator Values for `sp_trace_setfilter`

Value	Comparison Operator
0	= (equal)
1	<> (not equal)
2	> (greater than)
3	< (less than)
4	>= (greater than or equal)
5	<= (less than or equal)
6	LIKE
7	NOT LIKE

Fortunately, there is an easier way of generating a trace definition script. You can set up your traces by using the SQL Profiler GUI and script the trace definition to a file. When you have the trace defined and have specified the events, data columns, and filters you want to use, you select File, Export, Script Trace Definition. The SQL commands (including calls to the aforementioned system stored procedures) to define the trace, start the trace, and write the trace to a file are generated into one script file. You have the option to generate a script for either SQL Server 2000 or 2005. Listing 5.2 shows an example of a trace definition that was exported from the Profiler. It contains the trace definitions for the `TSQL` trace template. You must replace the text `InsertFileNameHere` with an appropriate filename, prefixed with its pathname, before running this script.

LISTING 5.2 A SQL Script for Creating and Starting a Server-Side Trace

```
/****************************************************/
/* Created by: SQL Server Profiler 2005             */
/* Date: 11/12/2006  09:55:24 AM         */
/****************************************************/

-- Create a Queue
declare @rc int
declare @TraceID int
declare @maxfilesize bigint
set @maxfilesize = 5

-- Please replace the text InsertFileNameHere, with an appropriate
-- filename prefixed by a path, e.g., c:\MyFolder\MyTrace. The .trc extension
-- will be appended to the filename automatically. If you are writing from
-- remote server to local drive, please use UNC path and make sure server has
-- write access to your network share

exec @rc = sp_trace_create @TraceID output, 0, N'InsertFileNameHere',
   @maxfilesize, NULL
if (@rc != 0) goto error
```

LISTING 5.2 Continued

```
-- Client side File and Table cannot be scripted

-- Set the events
declare @on bit
set @on = 1
exec sp_trace_setevent @TraceID, 14, 1, @on
exec sp_trace_setevent @TraceID, 14, 14, @on
exec sp_trace_setevent @TraceID, 14, 12, @on
exec sp_trace_setevent @TraceID, 15, 14, @on
exec sp_trace_setevent @TraceID, 15, 12, @on
exec sp_trace_setevent @TraceID, 17, 12, @on
exec sp_trace_setevent @TraceID, 17, 1, @on
exec sp_trace_setevent @TraceID, 17, 14, @on
exec sp_trace_setevent @TraceID, 11, 2, @on
exec sp_trace_setevent @TraceID, 11, 14, @on
exec sp_trace_setevent @TraceID, 11, 12, @on
exec sp_trace_setevent @TraceID, 13, 12, @on
exec sp_trace_setevent @TraceID, 13, 1, @on
exec sp_trace_setevent @TraceID, 13, 14, @on


-- Set the Filters
declare @intfilter int
declare @bigintfilter bigint

-- Set the trace status to start
exec sp_trace_setstatus @TraceID, 1

-- display trace id for future references
select TraceID=@TraceID
goto finish

error:
select ErrorCode=@rc

finish:
go
```

TIP

If you want to always capture certain trace events when SQL Server is running, such as auditing-type events, you can create a stored procedure that uses the `sp_trace` stored procedures to create a trace and specify the events to be captured. You can use the code in Listing 5.2 as a basis to create the stored procedure. Then you can mark the procedure as a startup procedure by using the `sp_procoption` procedure to set the `autostart` option. The trace automatically starts when SQL Server is started, and it continues running in the background.

Just be aware that although using server-side traces is less intrusive than using the SQL Profiler client, some overhead is necessary to run a trace. You should try to limit the number of events captured to minimize the overhead as much as possible.

Monitoring Running Traces

SQL Server 2005 provides some additional built-in user-defined functions to get information about currently running traces. Like the `fn_trace_gettable` function discussed previously, these functions return the information as a tabular result. The available functions are as follows:

- **`fn_trace_getinfo(`*`trace_id`*`)`**—This function is passed a `traceid`, and it returns information about the specified trace. If passed the value of `default`, it returns information about all existing traces. An example of the output from this function is shown in Listing 5.3.
- **`fn_trace_geteventinfo(`*`trace_id`*`)`**—This function returns a list of the events and data columns being captured for the specified trace. Only the event and column ID values are returned. You can use the information provided in Tables 5.2 and 5.3 to map the IDs to the more meaningful event names and column names.
- **`fn_trace_getfilterinfo(`*`trace_id`*`)`**—This function returns information about the filters being applied to the specified trace. Again, the column ID and logical and comparison operator values are returned as integer IDs that you need to decipher. See Table 5.4 for a listing of the column operator values.

LISTING 5.3 An Example of Using the Built-in User-Defined Functions for Monitoring Traces

```
SELECT * FROM ::fn_trace_getinfo(default)

traceid     property    value
----------- ----------- --------------------------------------------
1           1           2
1           2           C:\Program Files\Microsoft SQL Server\MSSQL.1\
                         MSSQL\LOG\log_375.trc
1           3           20
1           4           NULL
```

LISTING 5.3 Continued

```
1           5           1
2           1           0
2           2           c:\trace\mytrace.trc.trc
2           3           5
2           4           NULL
2           5           1


select * from ::fn_Trace_getfilterinfo(2)

columnid    logical_operator comparison_operator value
----------- ---------------- ------------------- -----------
3           0                0                   6
10          0                7                   Profiler
10          0                7                   SQLAgent
```

NOTE

You may be wondering why there is always a `traceid` with a value of 1 running when you run the `fn_trace_getinfo` procedure. This is the default trace that SQL Server automatically initiates when it starts. The default trace is enabled by default. You can identify which trace is the default by selecting from the `sys.traces` catalog view and examining the `is_default` column. The default trace captures a number of different types of events, including object creates and drops, errors, memory and disk changes, security changes, and more. You can disable this default trace, but it is generally lightweight and should be left enabled.

The output from the functions that return trace information is relatively cryptic because many of the values returned are numeric. For example, the property values returned by `fn_trace_getinfo` are specified as integer IDs. Table 5.5 describes of each of these property IDs.

TABLE 5.5 Description of Trace Property ID Values

Property ID	Description
1	Trace options specified in `sp_trace_create`
2	Trace filename
3	Maximum size of trace file, in MB
4	Date and time the trace will be stopped
5	Current trace status

Stopping Server-Side Traces

It is important to keep track of the traces you have running and to ensure that "heavy" traces are stopped. Heavy traces are typically traces that capture a lot of events and are run on a busy SQL Server. These traces can affect the overall performance of your SQL Server machine and write a large amount of information to the trace output file. If you specified a stop time when you started the trace, it will automatically stop and close when the stop time is reached. For example, in the SQL script in Listing 5.2, if you wanted the trace to run for 15 minutes instead of indefinitely, you'd set the value for the `stoptime` variable at the beginning of the script, using a command similar to the following:

```
set @stoptime = dateadd(minute, 15, getdate())
```

To otherwise stop a running server-side trace, you use the `sp_trace_setstatus` stored procedure and pass it the trace ID and a status of `0`. Stopping a trace only stops gathering trace information and does not delete the trace definition from SQL Server. Essentially, it pauses the trace. You can restart the trace by passing `sp_trace_setstatus` a status value of `1`.

Once you've stopped a trace, you can close the trace and delete its definition from SQL Server by passing `sp_trace_setstatus` the ID of the trace you want to stop and a status value of `2`. After you close the trace, you must redefine it before you can restart it.

If you don't know the ID of the trace you want to stop, you can use the `fn_trace_getinfo` function to return a list of all running traces and select the appropriate trace ID. The following is an example of stopping and closing a trace with a trace ID of `2`:

```
-- Set the trace status to stop
exec sp_trace_setstatus 2, 0
go

-- Close and Delete the trace
exec sp_trace_setstatus 2, 2
go
```

If you want to stop and close multiple traces, you must call `sp_trace_setstatus` twice for each trace. Listing 5.4 provides an example of a system stored procedure that you can create in SQL Server to stop a specific trace or automatically stop all currently running traces.

LISTING 5.4 A Sample System Stored Procedure to Stop Profiler Traces

```
use master
go
if object_id ('sp_stop_profiler_trace') is not null
    drop proc sp_stop_profiler_trace
go
```

LISTING 5.4 Continued

```
create proc sp_stop_profiler_trace @TraceID int = null
as

if @TraceID is not null
begin
    -- Set the trace status to stop
    exec sp_trace_setstatus @TraceID, 0

    -- Delete the trace
    exec sp_trace_setstatus @TraceID, 2
end
else
begin
--cg 2/8/06 - added a WHERE clause to eliminate a stop of the default trace
    declare c1 cursor for
    SELECT distinct traceid FROM :: fn_trace_getinfo (DEFAULT)
        WHERE traceId not in (select ID from sys.traces where is_default = 1)
    open c1
    fetch c1 into @TraceID
    while @@fetch_status = 0
    begin
        -- Set the trace status to stop
        exec sp_trace_setstatus @TraceID, 0

        -- Delete the trace
        exec sp_trace_setstatus @TraceID, 2
        fetch c1 into @TraceID
    end
    close c1
    deallocate c1
end
```

Profiler Usage Scenarios

This chapter has already covered many of the technical aspects of SQL Profiler, but what about some practical applications? Beyond the obvious uses of identifying what SQL statements an application is submitting, this section takes a look at a few scenarios in which the SQL Profiler can be useful. These scenarios are presented to give you some ideas about how SQL Profiler can be used. You'll see that the monitoring and analysis capabilities of SQL Profiler are limited only by your creativity and ingenuity.

Analyzing Slow Stored Procedures or Queries

After you have identified that a particular stored procedure is running slowly, what should you do? You might want to look at the estimated execution plan for the stored procedure, looking for table scans and sections of the plan that have a high cost percentage. But what if the execution plan has no obvious problems? This is when you should consider using the SQL Profiler.

You can set up a trace on the stored procedure that captures the execution of each statement within it, along with its duration, in milliseconds. Here's how:

1. Create a new trace, using the `TSQL_Duration` template.
2. Add the `SP:StmtCompleted` event from the stored procedure event class to the trace.
3. Add a filter on the `Duration` column with the duration not equal to `0`. You can also set the filter to a larger number to exclude more of the short-running statements.

If you are going to run the procedure from SSMS, you might want to add a filter on the `SPID` column as well. Set it equal to the process ID for your session; the SPID is displayed at the bottom of the SSMS window next to your username, in parentheses. This will trace only those commands that are executed from your SSMS Query Editor window.

When you run the trace and execute the stored procedure, you see only those statements in the procedure that have nonzero duration. The statements are listed in ascending duration order. You need to look to the bottom of the Profiler output window to find your longer-running statements. You can isolate these statements, copy them to SSMS, and perform a separate analysis on them to determine your problem.

You can also add showplan events to your Profiler trace to capture the execution plan as the trace is running. SQL Server now has showplan events that capture the showplan results in XML format. Traces with this type of XML output can have a significant impact on server performance while they are running but make the identification of poorly performing statements much easier. When tracing stored procedure executions, it is a good idea to add a filter on the specific stored procedure you are targeting to help minimize the impact on performance.

After you have run a trace with an XML showplan event, you can choose to extract the showplan events to a separate file. To do so, in the SQL Server Profiler you select File, Export, Extract SQL Server Events, Extract Showplan Events. At this point, you can save the showplan events in a single file or to a separate file for each event. The file(s) is saved with a `SQLPlan` file extension. This file can then be opened in SSMS, and the graphical query execution plan will be displayed.

Deadlocks

Deadlocks are a common occurrence in database management systems (DMBSs). In simple terms, deadlocks occur when a process (for example, SPID 10) has a lock on a resource that another process (for example, SPID 20) wants. In addition, the second process (SPID

20) wants the resource that the first process has locked. This cyclic dependency causes the DBMS to kill one of the processes in order to resolve the deadlock situation.

Resolving deadlocks and identifying the deadlock participants can be difficult. In SQL Server 2005 and past versions, trace flag 1204 can be set to capture the processes involved in the deadlock. The output is text based but provides valuable information about the types of locks and the statements that were executing at the time of the deadlock. In addition to this approach, SQL Server 2005 offers the ability to capture detailed deadlock information via the SQL Server Profiler. This type of tracing can be accomplished as follows:

1. Create a new trace, using a `Blank` template; this leaves the selection of all the events, data columns, and filters to you.
2. Add the `Locks:Deadlock graph` event to the trace from the `Locks` category. An additional tab appears on the Trace Properties window, named Event Extraction Settings.
3. Click the Save Deadlock XML Events Separately check box. This causes the deadlock information to be written to a separate file. You could also export the results after the trace has been run by using the File, Export option.

When you run this trace, it captures any deadlock event that occurs and writes it to the XML file specified. To test this, you can open two Query Editor windows and execute the following statements, in the order listed, and in the query window specified:

```
-- In Query Window # 1
--Step1
USE ADVENTUREWORKS
GO
BEGIN TRAN
   UPDATE HumanResources.Employee SET ModifiedDate = GETDATE()

-- In Query Window # 2
--Step2
USE ADVENTUREWORKS
GO
BEGIN TRAN
   UPDATE HumanResources.Department SET ModifiedDate = GETDATE()
   SELECT * FROM  HumanResources.Employee

-- In Query Window # 1
--Step3
   SELECT * FROM  HumanResources.Department
```

When the deadlock occurs, the results pane for one of the query windows contains a message similar to the following:

```
Msg 1205, Level 13, State 51, Line 3
Transaction (Process ID 55) was deadlocked on lock resources with another
process and has been chosen as the deadlock victim. Rerun the transaction.
```

When the row with the `Deadlock graph` event is selected in the Profiler output grid, a graph like the one shown in Figure 5.13 is displayed.

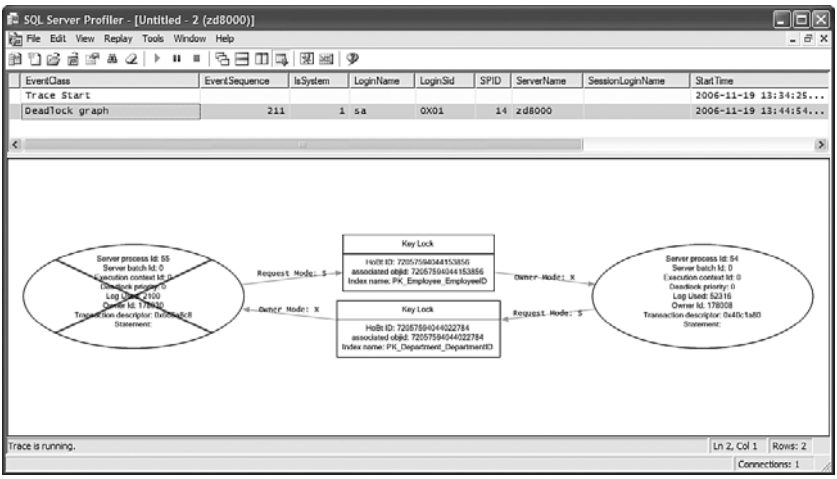

FIGURE 5.13 Output from the `Deadlock graph` event.

The `Deadlock graph` event contains a wealth of information about the deadlock occurrence. The oval nodes represent the processes involved in the deadlock. The oval with an X mark across it is the deadlock victim that had its process killed. The other oval represents the process that was allowed to complete when the deadlock was resolved. The boxes in the middle of the graph display lock information about the specific objects involved in the deadlock.

The graph is interactive and displays relevant information about the processes that were running when the deadlock occurred. For example, when you mouse over the oval nodes, pop-up text appears, displaying the SQL statement that was executing at the time of the deadlock. This is the same type of information that is displayed when the aforementioned trace flag is used, but the graph tends to be easier to decipher.

Identifying Ad Hoc Queries

One problem that can plague a production system is the execution of ad hoc queries against the production database. If you want to identify ad hoc queries, the application,

and the users that are running them, SQL Profiler is your tool. You can create a trace as follows:

1. Create a new trace, using the `SQLProfilerStandard` template.
2. Add a new `ApplicationName` filter with `Like Microsoft%`.

When this trace is run, you can identify database access that is happening via SSMS or Microsoft Access. The user, the duration, and the actual SQL statement are captured. An alternative would be to change the `ApplicationName` filter to trace application access for all application names that are not like the name of your production applications, such as `Not Like MyOrderEntryApp%`.

Identifying Performance Bottlenecks

Another common problem with database applications is identifying performance bottlenecks. For example, say that an application is running slow, but you're not sure why. You tested all the SQL statements and stored procedures used by the application, and they were relatively fast. Yet you find that some of the application screens are slow. Is it the database server? Is it the client machine? Is it the network? These are all good questions, but what is the answer? SQL Profiler can help you find out.

You can start with the same trace definition that was used in the previous section. For this scenario, you need to specify an `ApplicationName` filter with the name of the application that you want to trace. You might also want to apply a filter to a specific `NTUserName` to further refine your trace and avoid gathering trace information for users other than the one that you have isolated.

After you have started your trace, you use the slow-running application's screens. You need to look at the trace output and take note of the duration of the statements as they execute on the database server. Are they relatively fast? How much time was spent on the execution of the SQL statements and stored procedures relative to the response time of the application screen? If the total database duration is 1,000 milliseconds (1 second), and the screen takes 10 seconds to refresh, then you need to examine other factors, such as the network or the application code.

With SQL Server 2005, you also combine Windows System Monitor (Perfmon) output with trace output to identify performance bottlenecks. This new feature helps unite system-level metrics (for example, CPU utilization, memory usage) with SQL Server performance metrics. The result is a very impressive display that is synchronized based on time so that a correlation can be made between system-level spikes and the related SQL Server statements.

To try out this powerful new feature, you open the Perfmon application and add a new performance counter log. For simplicity, you can just add one counter, such as `% Processor Time`. Then you choose the option to manually start the log and click OK. Now, you want to apply some kind of load to the SQL Server system. The following script does index maintenance on two tables in the `AdventureWorks` database and can be used to apply a sample load:

```
USE [AdventureWorks]
GO
ALTER INDEX [PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID]
 ON [Sales].[SalesOrderDetail]
 REORGANIZE WITH ( LOB_COMPACTION = ON )
GO

PRINT 'FIRST INDEX IS REBUILT'

WAITFOR DELAY '00:00:05'

USE [AdventureWorks]
GO
ALTER INDEX [PK_Individual_CustomerID]
 ON [Sales].[Individual] REBUILD WITH
  ( PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
     ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON,
     SORT_IN_TEMPDB = OFF, ONLINE = OFF )
GO

PRINT 'SECOND INDEX IS REORGANIZED'
```

Next, you open the script in SSMS, but you don't run it yet. You open SQL Profiler and create a trace by using the `Standard Profiler` template. This template captures basic SQL Server activity and also includes the `StartTime` and `EndTime` columns that are necessary in order to correlate with the Perfmon counters. Now you are ready to start the performance log and the SQL Server Profiler trace. When they are running, you can run the sample load script. When the script has completed, you stop the performance log and the Profiler trace. You save the Profiler trace to a file and then open the file in the Profiler application.

The correlation of the Perfmon log to the trace output file is accomplished from within the Profiler application. To do this, you select File, Import Performance Data. Then you select the performance log file that was just created; these files are located by default in the `c:\perflogs` folder. Once you import the performance data, a new performance graph and associated grid with the performance counters is displayed in the Profiler, as shown in Figure 5.14

Now the fun begins! If you click one of the statements that was captured in the Profiler grid, a vertical red line appears in the Perfmon graph that reflects the time at which the statement was run. Conversely, if you click a location in the graph, the corresponding SQL statement that was run at that time is highlighted in the grid. If you see a spike in CPU in the Perfmon graph, you can click the spike in the graph and find the statement that may have caused the spike. This can help you quickly and efficiently identify bottlenecks and the processes that are contributing to it.

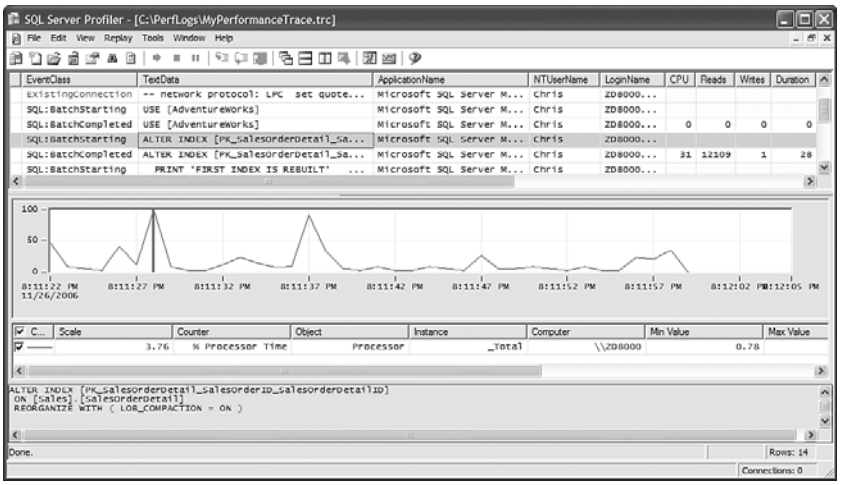

FIGURE 5.14 System Monitor counters correlated within a Profiler trace.

Monitoring Auto-Update Statistics

As discussed in Chapter 30, "Understanding Query Optimization," SQL Server updates index statistics automatically as data is changed in a table. In some environments, excessive auto-updating of statistics can affect system performance while the statistics are being updated. SQL Profiler can be used to monitor auto-updating of statistics as well as automatic statistics creation.

To monitor auto-updating of statistics, you create a trace and include the `AutoStats` event from the `Performance` event category. Then you select the `TextData`, `Integer Data`, `Success`, and `Object ID` columns. When the `AutoStats` event is captured, the `Integer Data` column contains the number of statistics updated for a given table, the `Object ID` is the ID of the table, and the `TextData` column contains names of the columns together with either an `Updated:` or `Created:` prefix. The `Success` column contains potential failure indication.

If you see an excessive number of `AutoStats` events on a table or index, and the duration is high, it could be affecting system performance. You might want to consider disabling auto-update for statistics on that table and schedule statistics to be updated periodically during non-peak periods. You may also want to utilize the new `AUTO_UPDATE_STATISTICS_ASYNC` database setting, which allows queries that utilize affected statistics to compile without having to wait for the update of statistics to complete.

Monitoring Application Progress

The 10 user-configurable events can be used in a variety of ways, including for tracking the progress of an application or procedure. For instance, perhaps you have a complex procedure that is subject to lengthy execution. You can add debugging logic in this procedure to allow for real-time benchmarking via SQL Profiler.

The key to this type of profiling is the use of the `sp_trace_generateevent` stored procedure, which enables you to launch the `User configurable` event. The procedure needs to

reference one of the `User configurable` event IDs (`82` to `91`) that correspond to the `User configurable` event `0` to `9`. If you execute the procedure with `eventid = 82`, then `User configurable` event `0` catches these events.

Listing 5.5 contains a sample stored procedure that (in debug mode) triggers the trace events that SQL Profiler can capture.

LISTING 5.5 A Stored Procedure That Raises `User configurable` Events for SQL Profiler

```
CREATE PROCEDURE SampleApplicationProc (@debug bit = 0)
as
declare @userinfoParm nvarchar(128)
select @userinfoParm = getdate()

--if in debug mode, then launch event for Profiler
--    indicating Start of Application Proc
if @debug =1
begin
       SET @userinfoParm = 'Proc Start: ' + convert(varchar(30),getdate(),120)
       EXEC sp_trace_generateevent @eventid = 83, @userinfo = @userinfoparm
end

--Real world would have complex proc code executing here
--The WAITFOR statement was added to simulate processing time
WAITFOR DELAY '00:00:05'

---if debug mode, then launch event indicating next significant stage
if @debug =1
begin
       SET @userinfoParm = 'Proc Stage One Complete: '
                           + convert(varchar(20),getdate(),120)
       EXEC sp_trace_generateevent @eventid = 83, @userinfo = @userinfoparm
end

--Real world would have more complex proc code executing here
--The WAITFOR statement was added to simulate processing time
WAITFOR DELAY '00:00:05' --5 second delay

---if debug mode, then launch event indicating next significant stage
if @debug =1
begin
       SET @userinfoParm = 'Proc Stage Two Complete: '
                           + convert(varchar(30),getdate(),120)
       EXEC sp_trace_generateevent @eventid = 83, @userinfo = @userinfoparm
end
```

LISTING 5.5 Continued

```
--You get the idea

GO
```

Now you need to set up a new trace that includes the `UserConfigurable:1` event. To do so, you choose the `TextData` data column to capture the `User configurable` output and any other data columns that make sense for your specific trace. After this is complete, you can launch the sample stored procedure from Listing 5.5 and get progress information via SQL Profiler as the procedure executes. You can accumulate execution statistics over time with this kind of trace and summarize the results. The execution command for the procedure follows:

```
EXEC  SampleApplicationProc @debug = 1
```

The resulting SQL Profiler results are shown in Figure 5.15.

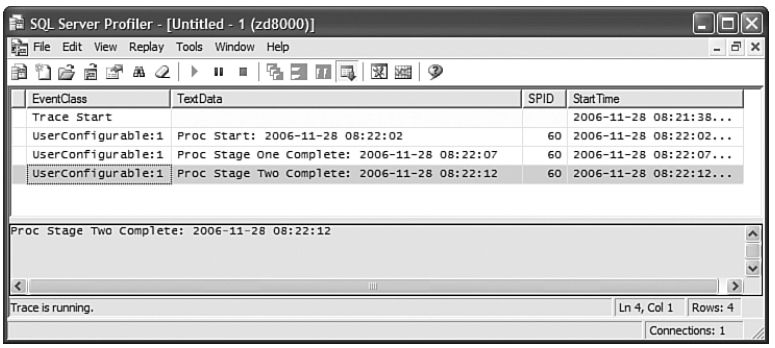

FIGURE 5.15 `User configurable` trace results.

There are many other applications for `User configurable` events. How you use them depends on your specific need. As is the case with many Profiler scenarios, there are seemingly endless possibilities.

Summary

Whether you are a developer or a database administrator, you should not ignore the power of the SQL Profiler. It is often one of the most underused applications in the SQL Server toolkit, yet it is one of the most versatile. Its auditing capabilities and ability to unravel complex server processes define its value.

This chapter wraps up the introduction to the tools and utilities available with SQL Server. Now you should be equipped to start administering and working with SQL Server.

The chapters in the next section focus on the overall administration of SQL, using some of the tools that you have been exposed to thus far. Chapter 6, "SQL Server System and Database Administration," gives you some insight into the inner workings of SQL Server and what it takes to effectively administer a SQL Server instance.

PART III

SQL Server Administration

IN THIS PART

This page intentionally left blank

CHAPTER 6

SQL Server System and Database Administration

IN THIS CHAPTER

This chapter outlines the role of a SQL Server system administrator and explores some of the methods that an administrator can use to query important system data. As with any other job, understanding the roles and responsibilities of an administrator is critical to doing the job well. You also need the right tools and the right information to do the job well and to do it efficiently.

The system data covered in this chapter provides some of the key information, and the methods discussed to access this information are some of the tools. System data discloses information that can be invaluable when assessing your SQL Server environment and is an essential part of administering a SQL Server database.

What's New in SQL Server System and Database Administration

The means for accessing system information has changed quite a bit in SQL Server 2005. Many of the system tables that you may have used in past versions are no longer system tables. You can still select from these tables (for example, `sysobjects`), but they exist in SQL Server 2005 as views. These views, which are discussed in much more detail later in this chapter, are called *compatibility views*. You can use compatibility views in much the same way as the system tables in past versions, but it is important to realize that they are now views. Microsoft has decided to hide from the SQL Server user the actual system tables that

these compatibility views represent. This presents some benefits for Microsoft because they can change the underlying system tables while still maintaining consistency for the end user through the views.

In addition to compatibility views, two other view types for selecting system information have been added to SQL Server 2005: catalog views and dynamic management views (DMVs). Catalog views provide access to metadata in the SQL Server environment. Using catalog views is the preferred method for accessing the metadata (as defined by Microsoft) and should be used instead of compatibility views in most instances. DMVs are a new tool for assessing the health of the SQL Server environment. DMVs provide a fast and easy method for getting diagnostic information about a server and can be used to provide information that may have been available in past versions only through system stored procedures, DBCC commands, or Profiler traces.

The great news about all these new views is that they can be used in a `SELECT` statement the same way as a table. Compare this to data that is returned in a system stored procedure: The system stored procedure provides valuable information, but you can't easily join the results to other tables or views, and you can't be selective with the data that is returned. The system views provide a great deal of flexibility and allow you easy access to a wealth of system-level information.

System Administrator Responsibilities

A system administrator is responsible for the integrity and availability of the data in a database. This is a simple concept, but it is a huge responsibility. Some large corporations place a valuation on their data as high as $1 million per 100MB. The investment in dollars is not the only issue; many companies that lose mission-critical data simply never recover.

The job descriptions for system administrators vary widely. In small shops, the administrator might lay out the physical design, install SQL Server, implement the logical design, tune the installation, and then manage ongoing tasks, such as backups. Larger sites might have tasks broken out into separate job functions. Managing users and backing up data are common examples of this. However, a lead administrator should still be in place to define policy and coordinate efforts.

Whether performed by an individual or as a team, the core administration tasks are as follows:

- Install and configure SQL Server.
- Plan and create databases.
- Manage data storage.
- Control security.
- Tune the database.
- Perform backup and recovery.

Another task that is sometimes handled by administrators is managing stored procedures. Because stored procedures for user applications often contain complex Transact-SQL (T-SQL) code, they tend to fall into the realm of the application developer. However, because stored procedures are stored as objects in the database, they are also the responsibility of the administrator. If your application calls custom stored procedures, you as the system administrator must be aware of this and coordinate with the application developers.

The system administration job can be stressful, frustrating, and demanding, but it is a highly rewarding, interesting, and respected position. As a system administrator, you are expected to know all, see all, and predict all, but you are well compensated for your efforts.

System Databases

SQL Server uses system databases to support different parts of the database management system (DBMS). Each database serves a specific role and stores information that SQL Server needs to do its job. The system databases are much like the user databases created in SQL Server. They store data in tables and contain views, stored procedures, and other database objects that you also see in user databases. They also have associated database files (that is, `.mdf` and `.ldf` files) that are physically located on the SQL Server machine. Table 6.1 lists the system database and the related database filenames.

TABLE 6.1 System Databases and Their Associated Database Files

Database	`.mdf` Filename	`.ldf` Filename
`master`	`master.mdf`	`mastlog.ldf`
`resource`	`mssqlsystemresource.mdf`	`mssqlsystemresource.ldf`
`model`	`model.mdf`	`modellog.ldf`
`msdb`	`msdbdata.mdf`	`msdblog.ldf`
`distribution`	`distmdl.ldf`	`distmdl.mdf`
`tempdb`	`tempdb.mdf`	`templog.ldf`

TIP

You can use the new `sys.master_files` catalog view to list the physical locations of the system database files as well as the user database files. This catalog view contains a myriad of information, including the logical name, current state, and size of each database file.

The folder that each of these database files is located in depends on the SQL Server installation. By default, the installation process places these files in a folder named `<drive>:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\`. You can move these files after the installation by using special procedures that are documented in the SQL Server Books Online topic "Moving System Databases."

The following sections describe the function of each system database.

The `master` Database

The `master` database contains serverwide information about the SQL Server system. This serverwide information includes logins, linked server information, configuration information for the server, and information about user databases that have been created in the SQL Server instance. The actual locations of the database files and key properties that relate to each user database are stored in the `master` database.

SQL Server cannot start without a `master` database. This is not surprising, given the type of information that it contains. Without the `master` database, SQL Server does not know the location of the databases that it services and does not know how the server is configured to run.

One change to the `master` database in SQL Server 2005 is the removal of system objects from the database. A new database, the `resource` database, now contains this information.

The `resource` Database

The `resource` database contains all the system objects that are deployed with SQL Server 2005. These system objects include the system stored procedures and system views that logically appear in each database but are physically stored in the `resource` database. Microsoft moved all the system objects to the `resource` database to simplify the upgrade process. When a new release of the software is made available, the updated system objects need only be updated in one spot—the `resource` database.

You do not see the `resource` database in the list of databases shown in SQL Server Management Studio (SSMS). For the most part, you should not be aware of the existence of the `resource` database. It has database files named `mssqlsystemresources.mdf` and `mssqlsystemresources.ldf` that are found in the same location as the `master` database files, but you cannot access the database directly. In addition, you do not see the database listed when selecting databases using system views or with system procedures, such as `sp_helpdb`.

The `model` Database

The `model` database is a template on which all user-created databases are based. All databases must contain a base set of objects known as the *database catalog*. When a new database is created, the model is copied to populate the requisite objects. Conveniently, objects can be added to the `model` database. For example, if you want a certain table created in all your databases, you can create the table in the `model` database, and it is then propagated to all subsequently created databases.

The `msdb` Database

The `msdb` database is used to store information for the SQL Server Agent, the Service Broker, Database Mail, log shipping, and more. When you create and schedule a SQL Server Agent job, the job's parameters and execution history are stored in `msdb`. Backups and maintenance plan information are stored in `msdb` as well. If log shipping is

implemented, critical information about the servers and tables that are involved in this process are stored in `msdb`.

The `distribution` Database

The `distribution` database is utilized during replication. It stores metadata and history information for all types of replication. It is also used to store transactions when transactional replication is utilized. By default, replication is not set up, and you do not see the `distribution` database listed in SSMS. However, the actual data files for the `distribution` database are installed by default.

Refer to Chapter 15, "Replication," for a more detailed discussion of the intricacies of replication.

The `tempdb` Database

The `tempdb` database stores temporary data and data objects. The temporary data objects include temporary tables, temporary stored procedures, and any other objects you want to create temporarily. The longevity of data objects in the temporary database depends on the type of object created. Ultimately, all temporary database objects are removed when the SQL Server service is restarted. The `tempdb` database is re-created, and all objects and data added since the last restart of SQL Server are lost.

`tempdb` can also be used for some of SQL Server's internal operations. Large sort operations are performed in `tempdb` before the result set is returned to the client. Certain index operations can be performed in `tempdb` to offload some of the space requirements. SQL Server also uses `tempdb` to store row versions that are generated from database modifications in databases that use row versioning or snapshot isolation transactions.

Maintaining System Databases

You should give system databases the same attention that you give your user databases. These databases should be backed up on a regular basis and secured in the event that one of them needs to be restored. All the system databases, with the exception of `tempdb` and the `resource` database, can be backed up. These same databases can also be restored to bring them back to a previous state.

It's important that you monitor the size of your system databases. The amount of data that accumulates in these databases can be significant. This is particularly true for the `tempdb`, `msdb`, and `distribution` databases. Large sort or index operations can increase the size of your `tempdb` database in a short period of time. The `msdb` and `distribution` databases contain a great deal of historical information. Take, for example, a server with hundreds of databases that have log backups occurring every 15 minutes. The information captured for each one of the backups is not significant, but the number of databases and the frequency of the backups causes many rows to be stored in the `msdb` database. Cleanup tasks and similar activities that remove older historical data can help keep the database size manageable.

System Tables

System tables contain data about objects in the SQL Server databases (that is, metadata) as well as information that SQL Server components use to do their job. Many of the system tables are now hidden and are no longer available for direct access by end users. In SQL Server 2005, compatibility views, which are discussed later in this chapter, have the same names as the system tables available in prior versions. For example, if you had a query in SQL Server 2000 that selected from `syscolumns`, this query continues to work in SQL Server 2005, but the results come from a view instead of a system table.

The system tables that you can view are now found in system databases, such as `msdb` or `master`. You can use the Object Explorer in SSMS to view the system tables in each database. Figure 6.1 shows the system tables listed for the `master` database in the Object Explorer:

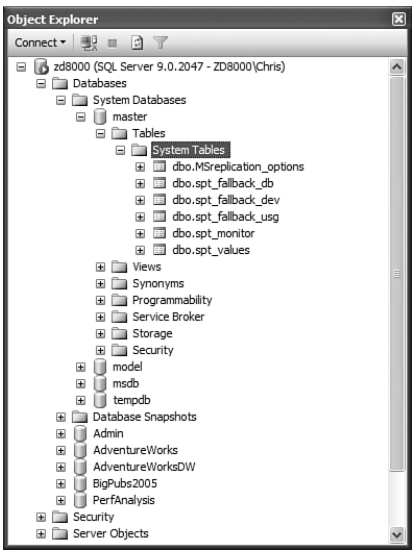

FIGURE 6.1 System tables listed in Object Explorer.

The most significant number of viewable system tables is found in the `msdb` system database. The system tables there support backup and restore, log shipping, maintenance plans, Notification Services, the SQL Server Agent, and more. You can retrieve a tremendous amount of information from these system tables if you know what you are looking for. The following example shows a query that selects from the system tables in `msdb` to report on recent restores for the `AdventureWorks` database:

```
select destination_database_name 'database', h.restore_date, restore_type,
   cast((backup_size/1024)/1024 as numeric(8,0)) 'backup_size MB',
   f.physical_device_name
```