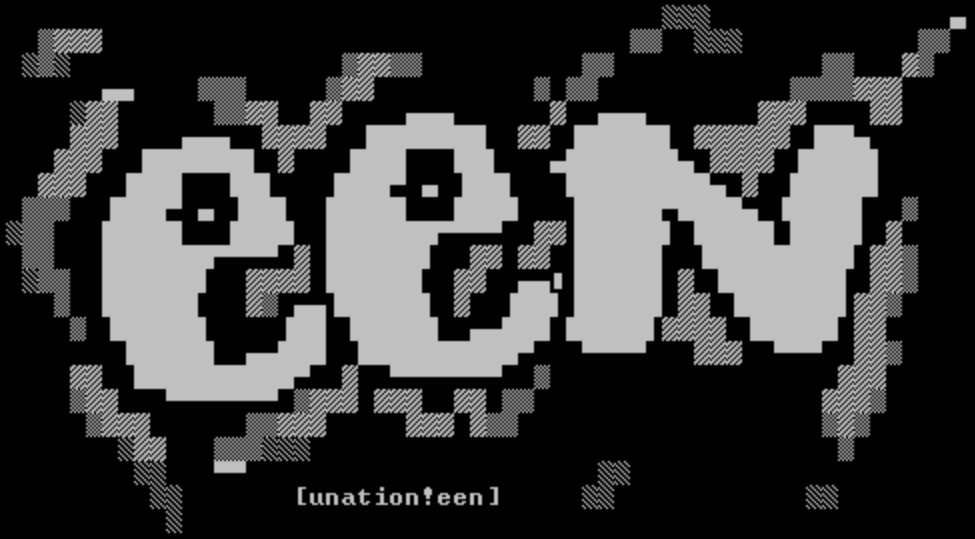


Enigma of Esoteric Nothingness

Designing SQL Server 2000 Databases for .NET Enterprise Servers
Introduction
Chapter 1: SQL Server 2000 Overview and Migration Strategies
Chapter 2: Installing and Configuring SQL Server 2000
Chapter 3: SQL Server Scalability and Availability
Chapter 4: Designing and Creating SQL Server Databases
Chapter 5: Database and Server Security
Chapter 6: Administration and Active Directory Integration
Chapter 7: SQL Server Backup and Recovery
Chapter 8: Microsoft English Query and Full-Text Search
Chapter 9: Importing and Exporting Data
Chapter 10: SQL Server Analysis Services
Chapter 11: Using XML with SQL Server
Chapter 12: Database Replication Techniques and Configuration
Chapter 13: Programming Tools and Technologies in SQL Server
Chapter 14: Performance-Tuning Tools and Techniques
Index
List of Figures
List of Tables
List of Sidebars



Designing SQL Server 2000 Databases for .NET Enterprise Servers
by Robert Patton and Jennifer Ogle

Designing SQL Server 2000 Databases for .NET Enterprise Servers

PUBLISHED BY
Syngress Publishing, Inc.
800 Hingham Street
Rockland, MA 02370

Copyright © 2001 by Syngress Publishing, Inc. All rights reserved. Printed in the United States of America. Except as permitted under the Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher, with the exception that the program listings may be entered, stored, and executed in a computer system, but they may not be reproduced for publication.

Printed in the United States of America

1 2 3 4 5 6 7 8 9 0

ISBN: 1-928994-19-9

Syngress Publishing, Inc., the author(s), and any person or firm involved in the writing, editing, or production (collectively “Makers”) of this book (“the Work”) do not guarantee or warrant the results to be obtained from the Work.

There is no guarantee of any kind, expressed or implied, regarding the Work or its contents. The Work is sold AS IS and WITHOUT WARRANTY. You may have other legal rights, which vary from state to state.

In no event will Makers be liable to you for damages, including any loss of profits, lost savings, or other incidental or consequential damages arising out from the Work or its

contents. Because some states do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

You should always use reasonable case, including backup and other appropriate precautions, when working with computers, networks, data, and files.

Syngress Media® and Syngress® are registered trademarks of Syngress Media, Inc. "Career Advancement Through Skill Enhancement™," "Ask the Author™," "Ask the Author UPDATE™," "Mission Critical™," and "Hack Proofing™" are trademarks of Syngress Publishing, Inc. Brands and product names mentioned in this book are trademarks or service marks of their respective companies.

KEY SERIAL NUMBER

001 58PPL99DSE

002 LSKDJ9878M

003 C3N44T8FQ7

004 KJ675HCC25

005 QCUCA94D26

006 PF62XD2G73

007 DT74HH52A4

008 LKJFARY343

009 65SKNSDAD5 010 6487FPS25N

Copy edit by: Darlene Bordwell

Technical edit by: Travis Laird

Project Editor: Maribeth Corona-Evans

Index by: Robert Saigh

Page Layout and Art by: Shannon Tozier

Co-Publisher: Richard Kristof

Distributed by Publishers Group West

Acknowledgements

We would like to acknowledge the following people for their kindness and support in making this book possible.

Richard Kristof, Duncan Anderson, Jennifer Gould, Robert Woodruff, Kevin Murray, Dale Leatherwood, Shelley Everett, Laurie Hedrick, Rhonda Harmon, Lisa Lavallee, and Robert Sanregret of Global Knowledge, for their generous access to the IT industry's best courses, instructors and training facilities.

Ralph Troupe and the team at Rt. 1 Solutions for their invaluable insight into the challenges of designing, deploying and supporting world-class enterprise networks.

Karen Cross, Kim Wylie, Harry Kirchner, John Hays, Bill Richter, Kevin Votel, Brittin Clark, Sarah Schaffer, Luke Kreinberg, Ellen Lafferty and Sarah MacLachlan of Publishers Group West for sharing their incredible marketing experience and expertise.

Mary Ging, Caroline Hird, Simon Beale, Caroline Wheeler, Victoria Fuller, Jonathan Bunkell, and Klaus Beran of Harcourt International for making certain that our vision remains worldwide in scope.

Anneke Baeten, Annabel Dent, and Laurie Giles of Harcourt Australia for all their help.

David Buckland, Wendi Wong, Daniel Loh, Marie Chieng, Lucy Chong, Leslie Lim, Audrey Gan, and Joseph Chan of Transquest Publishers for the enthusiasm with which they receive our books.

Kwon Sung June at Acorn Publishing for his support.

Ethan Atkin at Cranbury International for his help in expanding the Syngress program.

Joe Pisco, Helen Moyer, and the great folks at InterCity Press for all their help.

From Global Knowledge

At Global Knowledge we strive to support the multiplicity of learning styles required by our students to achieve success as technical professionals. As the world's largest IT training company, Global Knowledge is uniquely positioned to offer these books. The expertise gained each year from providing instructor-led training to hundreds of thousands of students worldwide has been captured in book form to enhance your learning experience. We hope that the quality of these books demonstrates our commitment to your lifelong learning success. Whether you choose to learn through the written word, computer based training, Web delivery, or instructor-led training, Global Knowledge is committed to providing you with the very best in each of these categories. For those of you who know Global Knowledge, or those of you who have just found us for the first time, our goal is to be your lifelong competency partner.

Thank your for the opportunity to serve you. We look forward to serving your needs again in the future.

Warmest regards,
Duncan Anderson
President and Chief Executive Officer, Global Knowledge

Contributors

Robert A. Patton (MCDBA, MCSD, MCSE+I, MCP+I) is a Software Engineer specializing in Microsoft Windows DNA applications using Interdev and Visual Basic with Windows 2000 and SQL Server. He is currently a Senior Applications Developer at PurchasingFirst.com in Dublin, OH and, in his position there, has done work for First Union National Bank, Corporate Strategic Services, the Midland Life Insurance Company, and Sykes Enterprises. Robert attended the University of Chicago, where he studied Public Policy. He earned his Bachelor of Science degree in Software Engineering from The Ohio State University. He lives in Dublin, OH with his wife Jenny and their sons Michael and Alex.

Jennifer Ogle (MCSE, MCP+I, MCNE, Oracle DBA) is Owner and President of Radioactive Frog Web Designs, Inc. (www.radfrog.com), which specializes in contract Web site design and development and database administration. Jennifer has over 20 years of experience in the IT industry, specializing in government, scientific, and manufacturing applications. Her recent assignments include SQL and Oracle development for Hoover Materials Handling Group and Lockheed Martin. In addition to her extensive software and hardware experience, she has programmed with Java, Visual Basic, and Visual InterDev, among many others. Jennifer lives with her husband Kent in Knoxville, TN.

Sean Campbell (MCSE, MCDBA, MCSD, MCT) is an Owner of 3 Leaf Solutions, LLC. He has worked in the education field for 7 years, with 5 years focused on IT training and consulting. Sean has worked with SQL Server since version 6. He has developed custom training and delivered MOC training for a number of years on SQL Server and other development technologies and has authored numerous Microsoft TechNet sessions on topics such as supporting SQL and upgrading to SQL Server 2000.

Sean's consulting experiences have run the gamut from database administration and application upgrades to .NET development. Sean has been working on projects with Microsoft since early Beta releases of SQL Server 2000 and has focused on SQL Server 2000's administration and implementation enhancements.

Mark Horninger (A+, MCSE+I, MCSD, MCDBA) is President and Founder of Haverford Consultants, Inc. (www.haverford-consultants.com), located in the suburbs of Philadelphia, PA. He develops custom applications and system engineering solutions, specializing primarily in Microsoft operating systems and Microsoft BackOffice solutions. Mark has over 12 years of computer consulting experience and has passed 26

Microsoft certified exams. During his career, Mark has worked on many extensive projects including database development using SQL 6.5, SQL 7, and SQL 2000; application development; training; embedded systems development; and Windows NT and 2000 project rollout planning and implementations. Mark lives with his wife Debbie and two children in Havertown, PA.

Steve Maier (MCSD) is a Senior Software Engineer for Heidelberg Digital L.L.C. (www.us.heidelberg.com), located in Rochester, NY. He develops custom PostScript drivers for Microsoft operating systems. Steve has 10 years of computer programming experience in DOS, Windows, and UNIX. During his career, Steve has done database development, manufacturing application development, driver development, and game development. He has also worked on FDA-approved medical software and has taught a variety of college classes including programming, system analysis, and PC repair. Steve lives with his wife Lisa and two children in Rochester, NY.

Henk-Evert Sonder (CCNA) has over 15 years of experience as an Information and Communication Technologies (ICT) professional, building and maintaining ICT infrastructures. In recent years, he has specialized in integrating ICT infrastructures with business applications and the security that comes with it. Currently, Henk works as a Senior Consultant for a large Dutch ICT solutions provider. His company, IT Selective, helps retailers get e-connected. Henk has also contributed to other Syngress books, including the *E-mail Virus Protection Handbook* (ISBN: 1-928994- 23-7).

Scott Delaney (MCSD, MCDBA, MCSE) is a Senior Consultant with TurnAround Solutions (www.turnaroundsolutions.com), a leading e-commerce consulting and product development firm based in Australia. TurnAround has a core client list including blue chips and multinationals and is known for its well-trained staff, processes, and quality development. In addition to his database work with SQL Server and Oracle, Scott also develops custom C++ and Java solutions. His current pet project is the development of SayIT, a full-featured speech synthesis package for use by people who have lost the ability to speak due to illnesses such as ALS or stroke. Scott lives in Tasmania, Australia with his wife Paula.

John Iwasz (MCSD) has over 5 years consulting experience developing custom applications. He specializes in n-tier and Web development using Microsoft technologies. John recently wrote an article on the IIS Metabase for *ASP Today* . During his career, John has worked on enterprise-wide projects as well as commercial development. His has designed and developed Web site front ends, middle tier components, and determined database architecture. John lives in Philadelphia, PA.

Melissa Craft (CCNA, MCSE, Network+, MCNE, Citrix CCA) is Director of e-Business Offering Development for MicroAge Technology Services. MicroAge is a global systems integrator headquartered in Tempe, AZ. MicroAge provides IT design, project management, and support for distributed computing systems. Melissa is a key contributor to the business development and implementation of e-business services. As such, she develops enterprise-wide technology solutions and methodologies focused on client

organizations. These technology solutions touch every part of a system's lifecycle—from network design, testing, and implementation to operational management and strategic planning.

Melissa holds a bachelor's degree from the University of Michigan and is a member of the IEEE, the Society of Women Engineers, and American MENSA, Ltd. Melissa currently resides in Glendale, AZ with her family, Dan, Justine, and Taylor. Melissa is the author of Syngress Publishing's best-selling *Managing Active Directory for Windows 2000 Server* (ISBN: 1-928994-07-5).

Cameron Wakefield (MCSD, MCP) is a Senior Software Engineer at Computer Science Innovations, Inc. (www.csihq.com) in Melbourne, FL, where he develops custom software solutions ranging from satellite communications to data-mining applications. His development work spans a broad spectrum including Visual C++, Visual Basic, COM, ADO, ASP, Delphi, CORBA, and UNIX. Cameron also develops software for a Brazilian hematology company as well as business-to-business Web applications. He also teaches Microsoft certification courses for Herzing College (AATP). He has passed 10 Microsoft certification exams. Cameron's formal education was in computer science with a minor in math at Rollins College. He lives in Rockledge, FL with his wife Lorraine and daughter Rachel.

Bret Stateham (CNE, MCSE, MCSD, MCDBA, MCT) is the Owner of Net Connex Technology Training and Consulting, L.L.C. (www.netconnex.com) in San Diego, CA. Bret's background includes extensive networking, development, and training experience. His most recent projects have involved custom training, database development and administration, XML schema development and standardization, custom application development, and countless hours developing Active Server Pages-based Web solutions. Bret and his wife Lori live in San Diego, CA.

Technical Editor and Contributor

Travis Laird (MCSE, MCDBA, A+, Network+, i-Net+, CIW) has extensive experience in a broad range of technology subjects ranging from network design and support through software design and development. Travis has worked on numerous e-commerce and line-of-business solutions based on Microsoft technologies including Windows 2000, Visual Basic, IIS, and SQL Server.

Travis currently works as an Independent Consultant in the New York City area. He recently designed and developed a Web-based e-commerce solution allowing a client to reach global markets with over 30,000 products. He attended Syracuse University and graduated with a Bachelor of Science degree in Finance and Management Information Systems. Travis lives in Westchester, NY and enjoys traveling with his ever-supportive fiancée Maria.

Solutions@syngress.com

With over 1,500,000 copies of our MCSE, MCSD, CompTIA, and Cisco study guides in print, we have come to know many of you personally. By listening, we've learned what you like and dislike about typical computer books. The most requested item has been for a web-based service that keeps you current on the topic of the book and related technologies. In response, we have created solutions@syngress.com , a service that includes the following features:

- A one-year warranty against content obsolescence that occurs as the result of vendor product upgrades. We will provide regular web updates for affected chapters.
- Monthly mailings that respond to customer FAQs and provide detailed explanations of the most difficult topics, written by content experts exclusively for us.
- Regularly updated links to sites that our editors have determined offer valuable additional information on key topics.
- Access to "Ask the Author"™ customer query forms that allow readers to post questions to be addressed by our authors and editors.

Once you've purchased this book, browse to www.syngress.com/solutions .

To register, you will need to have the book handy to verify your purchase.

Thank you for giving us the opportunity to serve you.

Introduction

As a consultant, the process of designing and developing technology-based solutions is one that you get to know intimately, thanks to iteration. Having been involved in a dozen or more projects in the past few years, I came to realize that a common component exists at the center of every one of those solutions. That component is the data storage system. Whether you are developing an electronic commerce Web site, a line-of-business solution such as a customer relationship management (CRM) application, or even a simple Win32 applet to store important notes, all of those solutions require a reliable, efficient, and proven data storage architecture. Providing that data storage component is the role of Microsoft SQL Server 2000. With industry-leading scalability, proven reliability, and unmatched ease of use, SQL Server 2000 can offer numerous advantages and efficiencies to building your solutions. Understanding the broad range of SQL Server's capabilities and how to implement them in your solutions is the goal of this book.

When Microsoft purchased the rights to the co-developed SQL Server project from Sybase in 1995, the necessity of a reliable data storage system was already evident. Released as Microsoft SQL Server, the product suffered for several years as a departmental database server, given its limited capabilities and scalability. SQL Server 7.0 entered the market in 1999 and moved SQL Server into the enterprise-class relational database management systems (RDBMS) market. With never before offered usability improvements, a self-tuning database engine, numerous application enhancements such as full-text indexing, English query capabilities, data analysis server, and a robust data access tier available in ActiveX Data Objects and OLE DB, SQL Server has gained ground on previous rivals such as Oracle, IBM DB2, and Sybase SQL Server. The results of several years of customer feedback, thousands of deployments, and billions upon billions of transactions, SQL Server 2000 is a market leader in RDBMS.

When I first started working with SQL Server in 1996, it was quite obvious from where its small-scale reputation had evolved. Clunky at best to work with and limited in its capabilities and management, SQL Server might have had a major version number, but it was still a 1.0 release. Over the next few years I worked with Microsoft SQL Server, Oracle, and Sybase SQL Server. Designers of software solutions at that time always looked to Oracle when it came to the question of where to store the data. In fact, other options were rarely even discussed. The release of SQL Server 7.0 was the result of significant reworking of the SQL Server product, which brought attention back to Microsoft in the enterprise-class RDBMS market.

What was significant about SQL Server 7.0 was its broad range of market usability. From small businesses with no dedicated technical staff to large multinational corporations with dozens of database administrators, SQL Server 7.0 was tailored to meet the demand. SQL Server finally found its way into the ranks of industry benchmarks for performance and scalability as well as usability. Those two categories are not typically mentioned in the same sentence; the ability to claim both characteristics is unique to Microsoft SQL Server.

In addition to this newfound status as an enterprise-class RDBMS, SQL Server 7.0 introduced numerous application features that were originally outside the scope of a native database server. Features such as Data Transformation Services (DTS) added native capabilities for importing and exporting data with heterogeneous systems, a task once left to custom-built utilities. An online analytical processing (OLAP) server engine opened the doors to data analysis capabilities once reserved for large corporations that had the resources to build and perform these activities. Smaller organization can use SQL Server 7.0 to take advantage of graphical wizards and automated maintenance routines to set up and manage their SQL Server solutions. Every organization can take advantage of the self-tuning capabilities that allow SQL Server to run at optimal levels right out of the box. These features and more opened up the market for SQL Server and positioned it to compete in the departmental segment, its previous niche, all the way to the enterprise-class RDBMS markets.

As with many Microsoft-based solution designers, I began working with SQL Server 7.0 while it was in beta and was immediately impressed with both the improvements to its relational database engine and its solution capabilities. Most of my previous SQL Server applications were migrated to SQL Server 7.0 within the first six months of its public availability. Under an aggressive schedule by most standards, the benefits of this feature-rich release were well beyond the concern over service packs, something Microsoft users have come to expect, and reliability testing proved its stability. New features such as native operating system file support replaced the “devices” storage model of previous versions, making database file management simpler. Data and transaction log file autogrow features reduce the amount of monitoring necessary to alleviate potential storage problems. Simplified setup of maintenance and backup tasks again minimize the amount of administration necessary. Features such as DTS replace and extend many of the custom import and export routines that I had developed and continued to maintain. The list of benefits to my SQL Server applications continues on.

Although the maturity period of SQL Server 7.0 was quick and many benefits are available from the simple upgrade from SQL Server 6.5 to 7.0, Microsoft’s development of SQL Server 2000 has a strong and interesting list of enhancements. New programming features—user-defined functions and table type variables as well as a redesigned Query Analyzer with functioning debugger—immediately attract the attention of any SQL developer. SQL Server 2000 offers a powerful list of feature enhancements both inside and outside. With nearly twice the amount of functional testing that SQL Server 7.0 was subject to, this latest version is the most scalable and reliable release of SQL Server to date. Designed with Windows 2000 Server in mind, SQL Server 2000 is scalable up to four-way clustering with support for up to 32 processors and 64GB of RAM. By any calculation, that is a powerful server configuration. On the more common side of database applications, enhancements in backup and recovery, DTS, and replication increase the usability and functionality of what have become common components in many database solutions.

This latest release of OLAP services, renamed Analysis Services, is an obviously mature version of Microsoft’s first take at OLAP in SQL Server. Analysis Services provide

greater integration and capabilities than their predecessor and now add a data-mining component for trend discovery in your data. Data mining is quickly becoming standard practice for organizations of any size as the competition for customers demands knowing who your customers are, what they purchase, and what they do at your Web site. The focus on Internet applications does not stop there. SQL Server 2000 is the first release with native support for HTTP and HTTPS protocols. Accessing and querying your SQL Server and Analysis Server databases is possible using standard Web protocols.

Aside from Web protocol support, Extensible Markup Language (XML) is at home in SQL Server 2000. The growing use of XML for nearly any imaginable application means that SQL Server 2000 is ready to support this new wave of solutions. The increasing trend of separating content from presentation in Web sites is much simpler with SQL Server's support for storing, delivering, and manipulating XML data. Other common practices of using XML to exchange data with heterogeneous systems, which once required complex conversion routines, is again natively possible with SQL Server 2000. This list of new and enhanced features is of tremendous interest and importance to the SQL Server application community and will surely continue to position SQL Server as an industry leader.

This book walks you through the SQL Server 2000 product so that you can take advantage of all that SQL Server has to offer. Building on the successful delivery of SQL Server 7.0, SQL Server 2000 presents integration and maturity of many features that were announced with and after SQL Server 7.0. SQL Server's rich integration with Internet technologies and native support for XML make it an exciting part of the next generation of solutions that many organizations are beginning to design in a growing global community. Enhancements in scalability and reliability position SQL Server 2000 as the first release to compete in the demanding markets of terabyte-size solutions and 24 x 7 availability.

With this exciting list of features in mind, we begin this book by reviewing the new and enhanced features of SQL Server 2000 and the migration considerations for those of you who are already using previous versions of SQL Server. The benefits and simplicity of migrating make this upgrade a practical necessity if your organization aims to compete in today's economy. A review of Microsoft's Distributed Internet Architecture (DNA) model and the pending evolution toward .NET will help you understand how SQL Server 2000 fits into Microsoft's vision of complete solutions design. From there, we review the installation options and procedures for SQL Server 2000. New features such as multiple-instance support allow you to run numerous distinct copies of SQL Server on a single server. This capability is a boon to the growing application service provider market, allowing you to partition large, expensive servers among multiple customers or applications, each with its own unique configuration. Several versions of SQL Server are available to meet both application requirements and budgets.

After we get SQL Server 2000 up and running, each chapter examines the capabilities available in SQL Server. From creating databases to setting up replication to using XML with SQL Server, we work through practical examples so you can begin using each of

these features in your application. Using both graphical wizards and Transact-SQL statements, you will learn how simple and powerful working with SQL Server can be. You will quickly discover that SQL Server 2000 is more than just a database server; it's a solution platform. Understanding SQL Server's capabilities will become critical to designing and delivering your applications in an increasingly demanding market.

Whether you are a veteran SQL Server user or you need to get SQL Server up and running for the first time, this book provides you with the insight and instruction of several authors, each with expertise in his or her area of SQL Server, so that you and your applications can begin enjoying the benefits of SQL Server 2000.

Chapter 1: SQL Server 2000 Overview and Migration Strategies

Introduction

Microsoft Structured Query Language (SQL) Server 2000 is the latest generation of the popular SQL Server product line and the second release since its core reengineering that produced SQL Server 7.0, released in 1999. This latest release of SQL Server adds native Extensible Markup Language (XML) support, enhanced online analytical processing (OLAP), data-mining capabilities, platform support for Windows 2000, integration with Windows 2000 Active Directory, and numerous performance, usability, and programming enhancements. SQL Server 2000 is available in six different editions to meet all levels of application development and delivery:

- SQL Server 2000 Enterprise Edition
- SQL Server 2000 Standard Edition
- SQL Server 2000 Personal Edition
- SQL Server 2000 Developer Edition
- SQL Server Desktop Engine (MSDE)
- SQL Server 2000 for Windows CE Edition

To meet scalability and availability goals, SQL Server 2000 is the first release designed and built to take advantage of Windows 2000 with support for up to 32 processors and 64GB of memory running on Windows 2000 Datacenter Server. Windows 2000 Active Directory integration adds enhanced server and security management features to SQL Server. The latest version of OLAP support, now called Analysis Services, provides numerous wizards for ease of use and the setup of OLAP and new data-mining solutions. One of the more publicized additions to SQL Server is its native support for XML. SQL Server 2000 offers support for storing, using, and updating XML documents—an important requirement because XML becomes the language of choice for many business systems and a fundamental architecture component of Microsoft .NET.

SQL Server 2000 is the first released member of the .NET Enterprise Server family and offers numerous advantages to organizations considering migration to this new platform. Enhancements in reliability, scalability, performance, and administration, along with strong compatibility with previous versions, make this release a strong candidate for early adoption into SQL Server 7.0 environments and an immediate migration from organizations running on SQL Server 6.5. Native XML support in SQL Server 2000 will help many organizations begin implementing this technology, which is quickly becoming a standard to both external and internal systems, including e-commerce, Web application services, and line-of-business applications.

This chapter discusses the changes in SQL Server 2000 as well as assists you in understanding the direction of Microsoft Windows Distributed interNet Applications

(DNA) Architecture Model toward .NET and the role of XML. You will review the available editions of SQL Server 2000 and their features and requirements so that you can choose the appropriate edition for your organization. The second half of this chapter discusses whether you should migrate to SQL Server 2000 as well as lays the groundwork for planning your migration.

Overview of SQL Server 2000: A .NET Enterprise Server

In September 2000, Microsoft officially announced its .NET Enterprise Server line and its commitment to .NET as Microsoft's application architecture model. The fundamental goal behind this new release of the company's popular server product line, now labeled .NET Enterprise Servers, is to provide simplified management, scalability, and availability throughout the enterprise, meeting the application goals of every organization and offering extensive support for .NET applications. SQL Server 2000 is the first .NET Enterprise Server available for public implementation and offers the data storage and management component of .NET services as well as a peek into the Microsoft's vision of .NET application capabilities.

Before we can see where the future will take us, it's always good to understand exactly where things came from. Microsoft SQL Server was first released as version 6.0 soon after Microsoft purchased and modified the code base to SQL Server from Sybase Corporation in 1995. Through version 6.5, released in 1996, SQL Server was accepted mainly as a departmental-scale database management system (DBMS) and lacked much of the scalability and reliability of enterprise-class solutions offered by companies such as Oracle and Informix. Administration of the SQL Server 6.0 and 6.5 products required knowledgeable SQL Server database administrators committed to monitoring server availability, activity, and performance. For SQL Server to have the broad market reach that Microsoft aims for in most of its products and to make it a fundamental component in its then-new Windows Distributed interNet Applications (DNA) Architecture Model, Microsoft needed to address the broad range of concerns and downfalls that plagued SQL Server's acceptance in both large and small organizations. In 1999, after several years of development and complete reengineering, Microsoft released SQL Server 7.0, which offered numerous enhancements in reliability, functionality, administration, security, performance, and scalability and allowed SQL Server to become the most popular relational database management system (RDBMS) in the market, with over 60 percent of all Web databases running on SQL Server by the end of 1999 and 70 percent of the total databases running on the Windows platform.

Soon after Microsoft launched SQL Server 7.0, several enhancements to the product, such as the XML Technology Preview, the OLAP Manager Add-In, and DTS Task Kit as well as two (now common) service pack updates, were released as downloadable additions. With the aggressive schedules and demanding markets that you and I represent, service packs are bound to continue, but all the previous additions and fixes have been refined and included in SQL Server 2000, along with countless enhancements in performance, availability, scalability, programmability, and management. SQL Server 2000 is light years ahead of the days of version 6.0 and is a required upgrade for nearly every SQL Server-based application.

The Future of Windows DNA: Microsoft.NET

In 1997, Microsoft announced the Windows Distributed interNet Applications (DNA) Architecture Model, which laid the foundations for building scalable, Web-based solutions based on Microsoft's line of servers, technologies, and development tools. Emerging from the popular, two-tier (or client/server) application architecture, DNA presents a distributed n-tier architecture model incorporating Web technologies such as Microsoft's Internet Information Services (IIS), including Active Server Pages (ASP), standard Web browser software such as Internet Explorer or Netscape Navigator, protocols including HyperText Transfer Protocol (HTTP), and enabling technologies such as Component Object Model (COM) and Data Access (ActiveX Data Objects, OLE DB). Via standard Web browser software, users are allowed access to DNA applications with little if any need for configuration of the client. Application services are centrally managed and delivered from the enterprise for enhanced reliability, scalability, and performance. This popular application architecture model accounted for roughly 40 percent of all secure, transacted Web-based applications built by the end of 1999.

The DNA Architecture Model is logically divided into three layers: Presentation Services, Business Services, and Data Services. Each layer plays a specific role in the application, as depicted in Figure 1.1 .

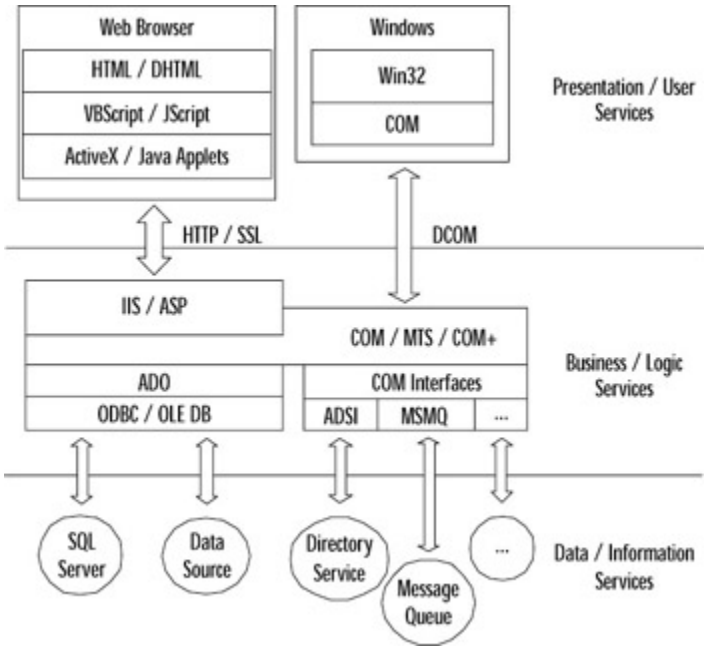


Figure 1.1: The DNA Architecture Model.

The Presentation Services layer encompasses the Web browser and client-side services such as HyperText Markup Language (HTML) to interpret and display Web pages, VBScript, and JavaScript for user input validation, Dynamic HTML (DHTML) for interface enhancements, and Java Applets or ActiveX Components for enhanced client-side functionality. The Business Services layer is responsible for much of the "work" in the application and handles tasks such as processing user input, applying business rules and application logic, validating data in and out of the Data Services layer, and delivering the application to the Presentation Services layer. The third layer of the DNA model is

Data Services, which is responsible for storing and managing all types of information, including databases, document storage, e-mail, and directory services data.

Now that you have a basic idea of the principles behind the DNA Model, we can discuss the Windows DNA Platform. The DNA Platform is the collection of software products, technologies, and tools that are used to physically build, host, and manage DNA applications. Having a conceptual model makes the theory of scalable, reliable, Web-based applications understandable, but to make it a reality, you need to be able to actually construct the layers to form your application. The Windows DNA Platform has continued to evolve as Microsoft releases new versions of existing products and adds entirely new product lines to the server, technology, and tools families. Today, the DNA Platform is made up of:

- Microsoft Windows NT 4.0/2000 Server
    - Internet Information Server/Services (IIS4/IIS5)
    - Message Queue Server (MSMQ)
    - COM/COM+
    - Data Access (ActiveX Data Objects, OLE-DB, ODBC)
    - Security Services
    - Network Load Balancing
    - Cluster Services
- Microsoft SQL Server 7.0
- Microsoft Exchange Server 5.5
- Microsoft SNA Server 4.0
- Microsoft Site Server 3.0 Commerce Edition
- Microsoft Visual Studio 6.0

Before we get into reviewing the responsibilities of each of these components, we should clarify how Windows DNA 2000 fits into all this. Windows DNA 2000 is the short-lived name for what is now called the Microsoft.NET Enterprise Servers. As Windows DNA evolves in to Microsoft.NET as the architecture model for scalable, reliable, Web-based applications and services, the Windows DNA Platform evolves into the .NET Enterprise Servers. The line of .NET Enterprise Servers consists of:

- **Microsoft Windows 2000 Server** Provides operating system and platform services such as storage management, security, Web services, messaging, and network services. Windows 2000 Server includes the following components:
    - Active Directory Services
    - Internet Information Services (IIS)
    - Active Server Pages+ (with the release of .NET)
    - Message Queue Server (MSMQ)
    - COM+
    - Data Access (ActiveX Data Objects+, OLE-DB, ODBC)
    - Security Service
    - Network Load Balancing
    - Cluster Services

- **Microsoft SQL Server 2000** RDBMS offers data storage, management, and analysis services. Provides XML support and Active Directory integration.
- **Microsoft Exchange Server 2000** Provides messaging and collaboration services such as e-mail, videoconferencing, and instant messaging. Provides Active Directory integration.
- **Microsoft Host Integration Server 2000** Offers access to legacy systems for information exchange, allowing integration with new technologies.
- **Microsoft Commerce Server 2000** Provides the services for implementing and managing electronic commerce Web sites on the Microsoft platform.
- **Microsoft BizTalk Server 2000** Offers the frameworks necessary to facilitate data communications between heterogeneous systems using standards-based formats such as XML.
- **Microsoft Application Center 2000** Provides the centralized management of distributed applications across multiple servers for scalability and reliability.
- **Microsoft Internet Security and Acceleration Server 2000** Provides enterprise-class firewall and Web cache for enterprise security and Web access performance.
- **Microsoft Mobile Information Server 2001** Although not scheduled to be available until 2001, Mobile Information Server supports delivering Wireless Application Protocol (WAP) and HTML to portable devices such as cellular phones and personal digital assistants (PDAs).
- **Microsoft Visual Studio.NET** Provides the tools and languages necessary to build applications using the .NET architecture components. The .NET programming architecture supported by Visual Studio.NET includes VB.NET, Active Server Pages+, ActiveX Data Objects+, and C# as well as C++. Missing from Visual Studio.NET is Visual InterDev. Microsoft Visual Studio.NET will incorporate a cross-language development environment with Web application integration throughout all Microsoft technologies, so the dedicated role of Visual InterDev is no longer necessary for Web programming.

Each of these server applications, technologies, and development tools plays a key role in delivering applications based on .NET. Microsoft has officially labeled .NET its vision for the next-generation Internet. What does all this mean for DNA developers and solutions built on the DNA platform? It means many exciting technology advancements and extensions to the original DNA architecture model.

What .NET is *not* is a replacement or shift away from the practiced and sound foundations of the DNA Architecture Model. .NET offers enhancements in many of the technologies with which developers currently work within their applications, including new versions of Active Server Pages, called ASP+ or Web Forms, and ActiveX Data Objects, called ADO+. Web Forms offer long-overdue validation services and advanced, server-side controls supporting events for building rich HTML-compliant application interfaces. Page logic is now separated from the HTML and compiled for better performance, and a common language runtime will allow developers to choose and mix their preferred languages with complete compatibility. Microsoft's commitment to Win32 applications continues with Win Forms, the Win32 counterpart to Web Forms. Although

the object models between the two are not the same, this unique approach to offering a comparable design model between Web applications and Windows applications will allow developers to migrate between platforms with a comfortable approach to writing code that has a higher level of productivity and developer availability.

All these enhancements are exciting for Web application designers and developers, but there is even more to .NET than the next version of these existing technologies. One of the core principles of .NET is the delivery of Web services to rapidly create powerful applications by integrating available services from within the organization and throughout the Internet. This web of interconnected applications and services is the core of .NET. To accomplish this web, .NET implements abundant use of XML and Simple Object Access Protocol (SOAP). SOAP and XML are both industry-standard technologies, which open the door for heterogeneous system communications. These new services can exist and run on any platform and be used by any application running on the platform of the developer's choice. Figure 1.2 illustrates the new vision in .NET.

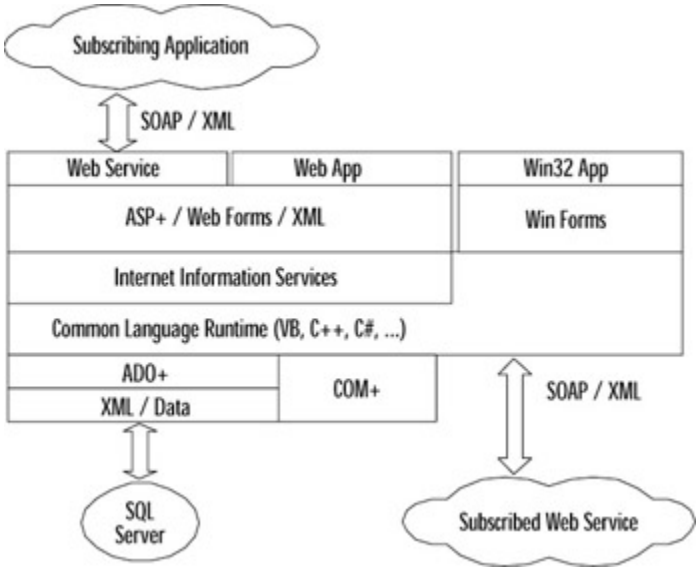


Figure 1.2: The .NET Framework Model.

The .NET Frameworks example extends the principles of DNA by providing integrated use of external services within the application, reducing application development and management time. Features such as Common Language Runtime leverage the broad range of developer skill sets, allowing complete support for mixed-language environments. Exchange of information with business partners and other organizations is possible using industry-standard technologies, XML, and SOAP. The ability even exists to expose and share components of the application with other organizations through Web services. Components such as order entry or product information are made available for integration into business partner systems and remote applications. .NET represents the next generation in distributed, reusable, and shared application architecture models.

So where does this leave SQL Server 2000 in the .NET world? As in nearly all applications, data storage and management are central to application functionality and

availability. SQL Server 2000 offers these traditional services and adds compelling new features such as native support for using and delivering XML documents, support for standard Internet protocols such as HTTP and Secure Sockets Layer (SSL) for secure Web access to SQL databases, and OLAP services, along with numerous scalability and performance enhancements. All these together make SQL Server 2000 the sole container of information in your existing DNA and future .NET applications.

New and Enhanced Features of SQL Server 2000

SQL Server 2000 delivers a more mature RDBMS from Microsoft. The first release since the near-entire redesign of SQL Server that resulted in version 7.0, SQL Server 2000 builds on that version and the feedback that resulted in two service packs of enhancements and fixes to the SQL Server architecture. This latest release offers enhanced reliability, scalability, programmability, and services for SQL programmers and application developers. Delivering key new features allows SQL Server to meet the demands of large-scale enterprise applications, including online transaction processing (OLTP), data warehousing, and electronic commerce, in which SQL Server continues to grow in market dominance.

Is .NET the End of DNA?

.NET is Microsoft's first foray into a new architecture model, designed around the increasing role of the Internet, since it announced DNA in the late 1990s. With the exception of some quick-to-guess assessments of early .NET information, no one is arguing that .NET will be the end of the popular and proven DNA Architecture Model. DNA's fundamental features, such as COM and multiple-tier environments, will continue to be invaluable in .NET solutions—but with a new twist: more efficient and rapid development and delivery. The fundamental principle behind .NET is a more productive development and runtime environment, an essential ingredient to meeting market demands and the increasingly rapid business shifts that the Internet has fueled.

.NET offers next-generation releases of popular technologies such as ASP+ and ADO+ in response to developer feedback and to support delivering powerful Web applications and services in record time. Unprecedented features such as mixed-language programming and a common-language runtime will allow organizations to leverage existing programming skills. Rich architecture services in the .NET Frameworks eliminate the need to repeatedly program and deal with tedious "plumbing" tasks such as memory or thread management and security. Support for nearly every popular language and richer architecture services are important features as organizations suffer from a documented and continued shortage of programmers.

.NET also offers a look into Microsoft's vision of software services. Although it will be some time before we see broad existence and acceptance of subscription-based software, the Web services model in .NET is the 1.0 release of this principle. The real benefit comes from the integration capabilities of services by allowing different groups to develop and manage features in which they are entrenched. By "subscribing" to these services, you can integrate them into your application and deliver a complete solution in record time.

If your organization has invested countless amounts of resources toward building DNA solutions, there is no need to begin sweating over .NET. Your experience with DNA and the ability to migrate existing applications and integrate .NET services into your DNA

solutions will make the process bearable and rewarding. What .NET brings to the table is that it enables your organization to deliver next-generation solutions with less time and money than was previously possible.

You can expect the .NET technologies to solidify in the coming months and become available for broad use in the first half of 2001. For more information, visit Microsoft's .NET Web site at www.microsoft.com/net .

As a member of Microsoft's .NET Enterprise Server family, SQL Server 2000 provides native support for XML as well as standard Internet protocols such as HTTP and SSL. Numerous productivity enhancements are welcome additions for SQL programmers, including new data types, trigger enhancements, user-defined functions, and a supercharged Query Analyzer that includes a built-in debugger that doesn't require godlike talents to configure! SQL Server 7.0's OLAP Services have "grown up" and been renamed Analysis Services, offering OLAP and data-mining capabilities native to SQL Server 2000. Having been designed for Windows 2000, SQL Server 2000 increases its scalability and availability levels, taking advantage of four-way fail-over clustering and support for up to 64GB of memory. A popular scalability enhancement in SQL Server 2000 is distributed partitioned views, which has allowed SQL Server to take over the Transaction Processing Council (TPC) leadership role in terms of price/performance measures and tremendously surpassing its rival, Oracle 8i, in scalability.

Sharing and exchanging data are common tasks in distributed application environments, and SQL Server 2000 shines in this area. Replication enhancements in SQL Server 2000 allow for queued updating subscribers and easier setup and management of replication solutions. Since version 7.0 of SQL Server, Data Transformation Services (DTS) have enjoyed many new fans, and this latest release adds a few of the "missing" pieces of the previous version.

If all this isn't enough to get you excited about this latest version of SQL Server, in the following sections we review the entire list of enhancements and additions to SQL Server 2000.

XML Support

XML, a subset of Standard Generalized Markup Language (SGML), is the latest addition to the arsenal of technologies available for building software solutions. Although not a new concept, the XML 1.0 recommendation was submitted to the World Wide Web Consortium (W3C) in 1998, many developers are finding it a strong and functional tool for communicating between heterogeneous systems and across the Internet. Microsoft has made such a strong commitment to XML that the company is touting it as a core technology component in its .NET architecture and services model. As its name implies, XML can be as simple or as complex as it needs to be to cater to a given situation, and its

HTML-like format makes it easy for new XML developers to quickly begin being productive with the technology.

Soon after Microsoft released SQL Server 7.0, the XML Technology Preview for SQL Server was released, providing insight into, and a draft of, what is now native XML support in SQL Server 2000. SQL Server 2000 offers native support for reading, writing, delivering, and using XML *documents* , the term for complete sets of XML tags and data. The following additions to SQL Server, along with the latest version of ActiveX Data Objects, version 2.6, provide complete support for using XML in your SQL Server-based applications:

SELECT…FOR XML New to Transact-SQL (TSQL) is the FOR XML statement clause. Addition of the FOR XML clause to a standard SELECT statement returns the results set as an XML-formatted document. The FOR XML clause offers three XML modes for formatting the resulting XML document: RAW, AUTO, and EXPLICIT. The XMLDATA option adds XML schema data to the result set so that column data types are accessible. The following example uses the FOR XML clause to return a formatted XML document, including data types, XMLDATA, from the Northwind sample database:

```
SELECT ProductName, UnitPrice, UnitsInStock, CompanyName FROM Products,
Suppliers WHERE Products.SupplierID = Suppliers.SupplierID AND
CompanyName='Tokyo Traders' FOR XML AUTO, XMLDATA
```

This example filters the results for one supplier, Tokyo Traders, so the results can be displayed here:

```
<Schema name="Schema6" xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">
<ElementType name="Products" content="eltOnly" model="closed"
order="many">
<element type="Suppliers" maxOccurs="*"/>
  <AttributeType name="ProductName" dt:type="string"/>
  <AttributeType name="UnitPrice" dt:type="fixed.14.4"/>
  <AttributeType name="UnitsInStock" dt:type="i2"/>
  <attribute type="ProductName"/>
  <attribute type="UnitPrice"/>
  <attribute type="UnitsInStock"/>
</ElementType>
<ElementType name="Suppliers" content="empty" model="closed">
  <AttributeType name="CompanyName" dt:type="string"/>
  <attribute type="CompanyName"/>
</ElementType>
</Schema>
<Products xmlns="x-schema:#Schema6" ProductName="Mishi Kobe Niku"
UnitPrice="97.0000" UnitsInStock="29">
  <Suppliers CompanyName="Tokyo Traders"/>
</Products>
<Products xmlns="x-schema:#Schema6" ProductName="Ikura"
UnitPrice="31.0000" UnitsInStock="31">
  <Suppliers CompanyName="Tokyo Traders"/>
```

```
</Products>
<Products xmlns="x-schema:#Schema6" ProductName="Longlife Tofu"
UnitPrice="10.0000" UnitsInStock="4">
  <Suppliers CompanyName="Tokyo Traders"/>
</Products>
```

OPENXML , **sp_xml_preparedocument** , **sp_xml_removedocument** In order to use and manipulate XML data in SQL Server, two new system-stored procedures and the OPENXML function have been added to SQL Server 2000. The OPENXML function returns a rowset of an XML document and can be used anywhere that a standard table, view, or OPENROWSET object is used to provide rowsets for processing. In order for SQL Server to process XML documents, the sp_xml_preparedocument stored procedure must be used to create the document object and return a numeric handle to the XML document. That handle is passed to and used by the OPENXML function to work with the XML document. After processing has been completed, the sp_xml_removedocument procedure must be used to destroy the XML document and release the server's resources. The following example inserts a new customer record in to the Customers table in the Northwind sample database using XML:

```
DECLARE @txtData   ntext
DECLARE @intXDoc   integer

SET @txtData = '<Customer custid="9etx3" companyname="Special Imports"
contactname="James Malcolm" contacttitle="Manager" address="555 Main
Street" city="Somecity" region="New York" zip="55555" country="United
States" phone="555-555-5555" fax="111-555-5555" />'

EXEC sp_xml_preparedocument @intXDoc OUTPUT, @txtData

INSERT INTO Customers(customerid, companyname, contactname,
contacttitle,
address, city, region, postalcode, country, phone, fax) SELECT * FROM
OPENXML(@intXDoc, '/Customer') WITH Customer

EXEC sp_xml_removedocument @intXDoc
```

XPath Queries, XML Views, and XDR Schemas The XML Path Language (XPath) Version 1.0 proposed specification states, "The primary purpose of XPath is to address parts of an XML document." Published in September 1999, XPath is the supported method of querying XML Views in SQL Server 2000. An XML View is an annotated XML Data-Reduced (XDR) Schema, which defines the structure of the XML document as well as element data types. The XDR Schema can then be modified to define the relationship between the XML document elements and the SQL database to create an XML view or mapping schema. The following simple example illustrates the structure of an XDR schema and an XML view. The changes between the XDR Schema and the XML View are highlighted in bold:

XDR Schema Example

```
<?xml version="1.0" ?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data">
 <ElementType name="Customer" >
  <AttributeType name="CustID" />
  <AttributeType name="Company" />
  <AttributeType name="Contact" />

  <attribute type="CustID" />
  <attribute type="Company" />
  <attribute type="Contact" />
 </ElementType>
</Schema>
```

XML View (Mapping Schema) Example

```
<?xml version="1.0" ?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data"

 xmlns:dt="urn:schemas-microsoft-com:datatypes"

 xmlns:sql="urn:schemas-microsoft-com:xml-sql"
>
 <ElementType name="Customer"
sql:relation="Customers"
>
  <AttributeType name="CustID" />
  <AttributeType name="Company" />
  <AttributeType name="Contact" />

  <attribute type="CustID"
sql:field="CustomerID"
/>
  <attribute type="Company"
sql:field="CompanyName"
 />
  <attribute type="Contact"
sql:field="ContactName"
 />
 </ElementType>
</Schema>
```

Note XPath is beginning to show its age as new specifications are being documented for querying XML data. The W3C, which manages many of the proposals and standards on Internet-related topics such as XML, hosts the XML Query Working Group, which is working toward a universal document data and query model. You can review these specifications on the W3C Web site (www.w3c.org). Microsoft also provides a developer area for XML technologies in Microsoft products at http://msdn.microsoft.com/xml/.

A discussion of XML support isn't complete without mentioning support for querying SQL Server 2000 using HTTP. Using the Configure SQL XML Support in IIS utility, located in the SQL Server program group, you can execute database queries against SQL Server using the Internet standard and firewall-friendly protocol, HTTP. By combining XML Views and XPath Query statements, you can

build data services to query and deliver XML information over the Web. The following URL example queries the XML View schema in the previous example for the customer record we inserted in the OPENXML example. Enter the following URL in the Address bar of Internet Explorer 5.0 or greater:

http:// localhost/nwind/customer.xml/Customer[@CompanyName="Special Imports"]

For this example to work, we need to configure IIS support for SQL using the Configure SQL XML Support in IIS utility, which is beyond the scope of this chapter. For additional information on configuring IIS to access SQL, please refer to Chapter 11 , "Using XML with SQL Server."

Development Tools and Technologies

Building SQL Server solutions is a demanding—and highly demanded—skill. The overall performance and scalability of your entire application can depend on a single T-SQL statement. Building efficient applications and delivering them in the least amount of time is both the goal of and the battle fought by many organizations; SQL Server 2000 brings many new tools and technologies that will allow you to meet the goals in your next SQL project. Every organization will find new tools in the SQL Server toolbox, from a charged-up Query Analyzer utility to database-level collation support.

Query Analyzer

The latest version of Query Analyzer, included with SQL Server 2000, includes many features for which developers have been patiently waiting. The first noticeable change is the included Object Browser, which provides a common tree view of every object in your SQL Server, ranging from tables and views down to functions and data types. This feature will be helpful for both experienced and new SQL developers. Instead of switching back and forth between Query Analyzer and Enterprise Manager or your data model, you can browse your database objects for column names, current indexes, parameters, and the like.

The right-click context menu in the Object Browser adds some interesting features: right-click on a table, and you can generate the table script or a complete SELECT statement; try a stored procedure and it will generate the alter code for modifying your procedure using the Query Analyzer. You will also notice a complete list of system functions. Browsing through the groups of functions, you will come across one or two that you could have used on that last project to save yourself several hours of development time. Before we move on to the next Query Analyzer feature, we should mention the second tab in the Object Browser, Templates. The Templates tab offers a complete array of T-SQL templates for everything from creating tables and using cursors to adding linked server logins. With the templates feature in the new Query Analyzer, you're bound to save some time you used to spend flipping through your reference books.

For those of us who have toiled with the SQL debugger capabilities of previous Microsoft development tools (and I'm talking about just getting them to work properly), this addition is long overdue. The included SQL Debugger in Query Analyzer makes stepping through your stored procedures as easy as it should have always been. If you installed the SQL Debugger component during installation, the right-click context menu of your stored procedures will include a Debug option. Select it, and a Debug Procedure dialog box will get you started by specifying your call parameters. Once in Debug mode, you will probably recognize the debugger tool from previous versions, but the bonus is that it is now integrated with Query Analyzer and doesn't require numerous Microsoft Knowledge Base articles to get it to work correctly.

Monitoring and tuning your SQL in Query Analyzer has also been stepped up with two additions to the Query menu, Show Server Trace and Show Client Statistics. Each of these options will step you through exactly what is happening when you execute your SQL, ranging from step activity to network round trips. The new Query Analyzer makes a strong showing, and with the exception of the SQL Debugger, these features work against SQL Server 7.0 database servers, so installing the latest client tools on your development workstation should be on your task list.

New Data Types

SQL Server 2000 adds three new data types to SQL Server's relational data-base engine: *bigint, sql_variant,* and *table* . The *bigint* data type is an 8-byte integer type with values ranging from -2^{63} (–9,223,372,036,854,775,808) through 2^{63-1} (9,223,372,036,854,775,807). The *sql_variant* data type is familiar to many Microsoft Visual Basic developers but is a new concept to SQL programmers. The *sql_variant* type is capable of storing various SQL Server data types, with the exception of *text, ntext, image, timestamp,* and *sql_variant* types. For example, you can store an integer or a varchar value in a table column with a *sql_variant* data type. To work with *sql_variant* data, a new function, named SQL_VARIANT_PROPERTY, has been created. The SQL_VARIANT_PROPERTY function can return the base type, precision, scale, collation, total bytes, or maximum length of the information stored in a *sql_variant* field or variable. These extended data properties are necessary to perform manipulations on the information or use it properly in your application.

Tip Although *sql_variant* is an interesting addition to SQL Server 2000, experience with Visual Basic development indicates that using variant type fields or variables can lead to unpredictability in your applications. Special rules apply when comparing variant type data, and unless you have a complete understanding of the nuances of using variants, testing and troubleshooting your application will leave you with an empty bottle of aspirin and a pounding headache.

Unless you find a specific need for which your application cannot predetermine the data type with which it will work (and I doubt this will be the case), it is advised that you steer clear of using *sql_variant* types. It's strong programming to type your variables and know where, when, and for what they will be used.

The most interesting of these three additions is the *table* data type. This virtual *table* variable can be used in place of traditional temp tables for building result sets or returning result sets from SQL Server 2000's new User-Defined Functions (UDF). Defining *table* type variables is similar to the traditional table definition structure, and once you have defined your *table* variable, you can select from, insert, update, and delete its contents using standard T-SQL statements. Its in-memory nature reduces locking and logging resources, and its local variable status offers a well-defined scope. For these reasons, the *table* data type is destined to become a popular addition for SQL developers.

Indexed Views

As one of its many performance enhancements, defining view indexes is now possible in SQL Server 2000. By creating an indexed view, the views result set is stored and indexed in the database for increased performance under usage scenarios that can take advantage of indexing. This new functionality can be implemented on existing database designs, so migrating a SQL Server 7.0 or 6.5 database to 2000 can take immediate advantage of indexed views without schema modifications.

Trigger Enhancements

Triggers are not new to SQL Server, but they have found their way into the list of updates in SQL Server 2000. SQL developers have been using triggers for years to automate and control activity against tables. SQL Server 2000 adds enhancements to the AFTER trigger (the only trigger mode in previous SQL Server versions) and offers a new trigger mode, the INSTEAD OF trigger. INSTEAD OF triggers can be defined for each of the data manipulation actions—UPDATE, DELETE, and INSERT—and can be created on tables or views. By replacing the action, INSTEAD OF triggers can be used to perform additional security checking, validate data, and even add update capabilities to views that could not previously support updating. As you might expect, only one INSTEAD OF trigger can be defined for each action.

SQL Server 7.0 added support for multiple-action triggers (AFTER mode), and SQL Server 2000 enhances multiple-trigger support with trigger priorities for AFTER triggers. Using the sp_settriggerorder system-stored procedure, you can specify the first and last triggers to fire. Although the remaining triggers are fired randomly between the first and last assigned triggers, this new level of control can be useful in scenarios in which notifications or pre- and post-processing is necessary.

Referential Integrity Enhancements

Before SQL Server 2000, enforcing referential data integrity involved creating UPDATE and DELETE triggers to manage the data between referenced tables. Without enforcing these rules, for example, you could end with an Orders table that referenced products that no longer exist in the Products table because product identifiers were changed or product records were deleted. Referential integrity enhancements now allow you to define column references and action responses when the table is created. Specifying the referenced table

and column along with the action and response greatly simplifies the process of implementing cascading updates and deletes. Instead of creating a table and then creating an UPDATE and DELETE trigger on the table to handle referential integrity constraints, the following example completes the entire task in the CREATE TABLE statement, allowing SQL Server to automatically handle the update and delete actions:

```
CREATE TABLE RTable
         (RTableID bigint PRIMARY KEY,
         ColumnOne bigint,
         ColumnTwo datetime,
         FWTableID bigint
            FOREIGN KEY REFERENCES WTable(WTableID)
                ON DELETE CASCADE
                ON UPDATE NO ACTION
         )
```

User-Defined Functions

A significant addition to the programmability of SQL Server is the new User-Defined Function (UDF) object. We use dozens of the built-in SQL functions to manipulate and process data, define computed columns, and control logic flow in all our SQL applications, but now SQL development is no longer restricted to that list of predefined SQL functions from Microsoft. With support for returning single values such as integer or character data and the ability to return the new *table* data type, UDFs in SQL Server will increase developer productivity and code reuse in SQL applications.

The process of creating UDFs is similar to creating other objects in SQL Server. The CREATE FUNCTION statement allows you to define input parameters and the return parameter type of your custom function. The following example creates a new function to return a *table* data type that contains the product name and unit price from the Products table in the sample North-wind database for all products that have a "units in stock" greater than the input parameter—essentially, a parameterized view. This function can be used in the FROM clause of a T-SQL statement as a rowset:

```
CREATE FUNCTION GetProductsAvail ( @intStockQty int )
RETURNS @ProductList TABLE
(
ProductName   nvarchar(40),
UnitPrice           money
)
AS
BEGIN
     INSERT INTO @ProductList
          SELECT productname, unitprice
          FROM Products
          WHERE UnitsInStock > @intStockQty
     RETURN
END
```

Index Enhancements

The indexing engine in SQL Server 2000 has been enhanced to provide more efficient index creation as well as additional indexing options. Several changes to the way SQL Server builds indexes provide noticeable performance improvements in large indexing tasks. The first of these enhancements is the support for parallel scanning and sorting during index creation, which is available on multiprocessor servers. This multi-threaded approach has obvious advantages in speeding up index creation. The CREATE INDEX statement now includes a SORT_IN_TEMPDB option to allow using the tempdb to store sort results during the index-building process. When the tempdb database is located on a separate disk system from the filegroup, the default location for intermediate sort results, the additional disk I/O performance can offer noticeable performance advantages during index building. In addition to this performance improvement, isolating the read activity to the filegroup and the write activity to the tempdb delivers a more serial approach to writing the sort results and the eventual index, creating a more contiguous physical index.

Aside from being able to create indexes on views, as mentioned previously, SQL Server also supports defining indexes on computed columns. The numerous improvements in index creation and index options improve SQL Server's relational database engine.

Note A useful addition to the Database Console Commands (DBCC) is the INDEXDEFRAG statement. New to SQL Server 2000, DBCC INDEXDEFRAG defragments both clustered and secondary indexes. For clustered indexes, this means physically resorting your data in logical order, which can improve scan performance. In addition to defragmenting table and view indexes, the INDEXDEFRAG statement compacts index pages according to the index fill factor, removing unused index pages.

Shared Session Information

In the world of Web application development, session state has become a popular issue, so many of us are aware of concepts behind session-level information, its benefits, and its pitfalls. SQL Server now offers a similar concept by allowing the storage of up to 128 bytes of binary information in the context of the current connection. This information is stored in the sysprocesses table by the process ID and is available across multiple batches, including stored procedures, triggers, and UDFs. The ability to store information accessible to any routine running under the current connection will allow you to control application logic based on previous activity.

To store session information for the current connection, SQL Server provides the SET CONTEXT_INFO statement, which accepts a single parameter of up to 128 bytes of binary data. You can retrieve this information in any routine under your current connection by selecting the context_info column from the sysprocesses table, where the spid is the system variable @@SPID.

Collation Support

We have all heard it a thousand times: The Internet has made the world a smaller place. Although that is true, we still don't all read and write the same language or use the same alphabet. For SQL Server to understand and store the proper form of information that it receives, we must select a *collation* . Collations are the rules by which SQL Server sorts, compares, and stores data. Selecting a collation for SQL Server is not a new concept; we have always had to specify the sort order and code page during the setup process, and this continues to be the case in SQL Server 2000. This latest release of SQL Server, however, takes it one step further by allowing you to specify the collation property at both the database and column levels. Organizations such as application service providers (ASPs) or global firms that have to deal with multiple languages and alphabets can now service different regional applications or customers on a single SQL Server 2000 installation. This approach takes advantage of SQL Server's scalability and simplifies server management by reducing the number of installations necessary to meet application requirements.

Extended Properties

Many smaller database applications such as Microsoft Access have been using extended properties for some time. *Extended properties* allow the developer to define additional descriptive information for database objects and object components such as table columns or indexes. This additional information can be used by client applications or developers to understand the object and to specify characteristics such as the display format for a character column. SQL Server provides three system-stored procedures, sp_addexptendedproperty, sp_dropextendedproperty, and spupdateextendedproperty, and one system function, fn_listextendedproperty, to manage and read extended properties.

Meta Data Services

New to SQL Server 2000, Meta Data Services incorporate and extend the Microsoft Repository engine for storage and management of metadata. *Metadata* is a commonly used term for descriptive information about applications and information systems. Many development tools can use metadata to define and exchange information about the application and components with other modeling or development tools. Meta Data Services in SQL Server support the Open Information Model (OIM), which is a set of standard object models that can be used to exchange common format metadata with other systems that can process and use metadata.

A new Meta Data Browser will allow you to view the repository database contents. The repository engine in Meta Data Services has been enhanced and extended to offer additional programmability and modeling features. A new COM object model supports features such as creating repository views for querying the repository database as well as enhancing performance and functionality.

Analysis Services

Online Analytical Processing (OLAP) Services are an increasingly popular component of SQL Server; the latest enhancements offer improved OLAP features as well as new data-mining capabilities that are extremely useful for discovering data trends. Analysis Services in SQL Server 2000 offer simplified configuration and an enhanced feature set over the previous version of OLAP Services included with SQL Server 7.0. Numerous enhancements and new features increase Analysis Services usability, performance, and security over the previous version. Additions such as new cube and dimension types provide enhanced OLAP analysis capabilities. The new data-mining capabilities offer an entirely new level of analysis that has the ability to find trends in both OLAP and relational data sources.

Analysis Services include several usability enhancements, such as Windows 2000 Active Directory integration. Analysis servers can publish their status in Active Directory, allowing enterprise users to locate and return information about active analysis servers. Analysis Services counters are included with this release, offering support for Windows Performance Monitor. Using Performance Monitor, authorized users can monitor performance and activity levels of the analysis server. Support for multiple administrators with active object locking will offer increased server use through availability in the enterprise.

Additional Analysis Services enhancements include:

- **Integrated backup and object copy and paste functionality** The OLAP database backup and restore as well as object copy and paste features were previously released as add-ins to OLAP services. These features have now been integrated into the Analysis Manager.
- **Multidimensional expressions (MDX)** Aside from additional MDX functions in this version of Analysis Services, the new MDX Builder utility is available for building MDX statements using drag-and-drop techniques. With the increased use of Analysis Services and the complexities of MDX syntax, the MDX Builder tool will be a welcome addition for many users.
- **Security enhancements** Using role-level security in Analysis Services, administrators can set read- and read/write-level permissions on individual dimensions, levels, and members as well as specific cube cells.
- **Analysis Server accessibility** The ability to communicate with Analysis Services through IIS over the Internet and through firewalls is available in this release, with support for the HTTP and HTTPS protocols.

OLAP Enhancements

SQL Server 7.0 successfully delivered Microsoft's first OLAP product, OLAP Services. This tremendously popular feature opened the doors for many organizations to the capabilities and value of OLAP without the need for statisticians. Its ease of use allows smaller organizations that lack dedicated data warehousing and OLAP designers to begin building OLAP solutions for analyzing information in their organizations. The latest

release of OLAP in Analysis Services brings with it numerous performance, scalability, and feature enhancements.

Performance and scalability enhancements in the OLAP engine include support for distributed partitioned cubes and linked cubes. Distributed partitioned cubes allow for distributing cube data among multiple remote analysis servers while providing a central analysis server for cube administration. To eliminate duplicate copies of cube data across the enterprise, linked cubes can be used so that access to cube data is available to multiple analysis servers without managing multiple copies of the information. Real-time OLAP (ROLAP) in Analysis Services offers an OLAP view of frequently changing data through automatic notification of data changes by SQL Server 2000. Cube processing enhancements, drill-through support, calculated cells, and hidden elements offer more responsive, configurable, data rich cubes.

The new Virtual Cube Editor replaces the Calculated Member Manager Add-In offered with SQL Server 7.0 OLAP Services and allows editing of virtual cubes with a tool similar to the Cube Editor. Actions, new to Analysis Services, offer user response capabilities to cubes. By defining actions, an end user can initiate an operation such as launching an external application or sending a notification based on the selected cube or portion of a cube as the action parameter. This functionality can be used to initiate business processes based on analysis results.

Extending the capabilities of Analysis Services, several new dimension types have been added. They include:

- **Changing** Allows dimension members to be deleted, added, renamed, or moved without reprocessing the cube.
- **Dependent** Specifies that members are determined by members of another dimension. This dimension type offers advantages when the nonintersecting member combinations offer a high percentage of combinations that cannot coexist.
- **Parent/child** Supports hierarchies based on parent/child links between members in columns in the source tables such as an Employee table, which contains an employee ID and a manager ID that links to the employee ID of another employee who acts as that person's manager. This relationship results in an organizational chart showing the employee/manager hierarchy.
- **Ragged** Has one or more members whose logical parents do not exist in the level immediately above the member.
- **ROLAP** Using the ROLAP storage mode, supports extremely large dimensions that are not subject to the size limitations of multidimensional OLAP (MOLAP).
- **Write-enabled** Can be updated by the Analysis Manager and client applications that support write-back.

Data Mining

Data-mining capabilities in Analysis Services open the door to a new world of analysis and trend prediction. By discovering trends in either relational or OLAP cube data, you can gain a better understanding of business and customer activity, which in turn can drive more efficient and targeted business practices. Based on algorithms created by Microsoft Research, data mining can analyze and present a grouping and predictive analysis of your data source. Microsoft Research has created two algorithms for building data-mining models that are included in Analysis Services:

- **Decision trees** A decision tree results in a tree structure classification by which each node in the tree represents a question used to classify the data. For example, a decision tree analysis might be used to determine who is most likely to purchase a particular type of product on the Web. Decision tree analysis is a widely used technique for statistical analysis.
- **Clustering** Clustering is another popular analysis technique and is based on grouping records into neighborhoods or clusters based on similar, predictable characteristics.

Analysis Services also support algorithms developed by third parties.

Data-mining algorithms can also be used by Analysis Services client applications to build data-mining analysis models on OLAP cubes and for incorporating mining results into OLAP cubes. Extensions to MDX offer data-mining capabilities in connection with OLAP cubes. The Data Trans-formation Services (DTS) task for Analysis Services has been enhanced to support mining model processing, and the new Mining Model Prediction Task is available to support creating predictions in DTS packages.

Building and managing data-mining models in SQL Server 2000 Analysis Services is possible via several wizards and editors for increased usability. The Mining Model Wizard will walk you through the process of setting up your data-mining project. The Mining Model Editor is used to modify your data-mining model, and the Mining Model Browser can graphically display the results of your mining model analysis. The simplicity of these tools, combined with the rich functionality of data mining in Analysis Services, will make SQL Server an even stronger player in the business intelligence arena and offer a complete data analysis solution for Windows DNA and .NET solutions.

SQL Server Administration

SQL Server 2000 continues to offer centralized administration tools, including its Enterprise Manager Snap-In for the Microsoft Management Console (MMC), Microsoft's enterprise management framework. The Enterprise Manager in SQL Server 2000 will be familiar to previous users of the tool. Other than minor aesthetic changes, such as a modified Taskpad view, your first impression will be that it is nearly identical to its previous version. You can even use the SQL Server 2000 Enterprise Manager to register and administer SQL Server 7.0 servers. Under the hood, though, Enterprise Manager has added a few options to support new functionality in SQL Server 2000. You can now attach and detach SQL databases from the right-click context menu, as well as issue

shrink commands to free unused space. To support Analysis Services, an additional MMC Snap-In is available for administering local and remote analysis servers. You can additionally register previous OLAP servers from SQL Server 7.0 for centralized administration of both analysis and OLAP servers throughout your enterprise.

SQL Server's self-tuning mechanisms have been enhanced in this version with the addition of several new dynamic algorithms. These new algorithms provide optimization in environments that take advantage of new hardware technologies such as large amounts of memory, storage area networks, system area networks, and ultra-high-speed disk subsystems. In addition, several new counters have been added to SQL Profiler for monitoring data and log file growth. SQL Profiler can be used to monitor and record server activity and automated management tasks.

SQL Server offers new tools to simplify routine tasks such as server setup and database migrations. The new Copy Database Wizard can be used to transfer databases between servers for migrating or upgrading from SQL Server 7.0. Databases and their shared objects, logins, and error messages can be moved or copied to the new destination server. Transfer jobs can even be set up and scheduled to execute at a later time.

Windows 2000 Active Directory Integration

SQL Server 2000 offers support for Windows 2000 Active Directory (AD). With AD integration, SQL Servers can be registered in AD, offering physical server location independence for applications that connect to and use SQL Server databases, publications, or Analysis Services. By developing Active Directory Service Interfaces (ADSI) enabled applications, custom applications can locate SQL Server instances and connect to or report on instances as well as offerings such as publications and individual data-bases, which must individually be registered in AD. You can register servers, databases, and publications in AD from the objects' Properties dialog box in Enterprise Manager or using T-SQL system procedures. The transparency offered through AD registration allows databases to be moved between servers or instances without crippling the applications and services that use them.

SQL Server 2000 uses the following Active Directory schema objects:

- **MS-SQL-SQLServer** SQL Server instance.
- **MS-SQL-SQLPublication** SQL Server replication publication.
- **MS-SQL-SQLDatabase** SQL Server database instance.
- **MS-SQL-OLAPServer** SQL Server Analysis Server instance.

Scalability and Availability

One of Microsoft's primary goals in delivering SQL Server 2000 was to improve on the scalability and availability features of the company's immensely successful version 7.0. Until SQL Server 7.0, Microsoft's relational database server had seen little activity in the area of large, enterprise-scale applications. Version 7.0 made tremendous headway in

establishing Microsoft's presence in the enterprise-class RDBMS arena. SQL Server 2000 improves on that success by delivering a benchmark winning database solution, capturing the Transaction Processing Council's (TPC) TPC-C benchmark record in October 2000 with the first score above the half-million mark, at 505,302 transactions per minute (tpmC). As of the October 2000 benchmarks, SQL Server 2000 holds four of the five top scores on the TPC-C benchmarks, which measure throughput in business transactions per minute, along with performance and scalability. These benchmarks were attainable through SQL Server 2000's new scale-out capabilities. By leveraging inexpensive, commodity hardware and database partitioning, SQL Server is able to distribute activity across multiple servers with distributed partitioned views.

To accomplish these goals and position SQL Server 2000 as the leading RDBMS in terms of performance, price, and scalability, Microsoft has delivered significant enhancements to SQL Server in the areas of capacity, scalability, and availability.

Scalability Enhancements

As the first version of SQL Server designed for Windows 2000, SQL Server 2000 is able to take advantage of Windows 2000 Server's features and platform stability. Running on Windows 2000 Datacenter Server, SQL Server 2000 is able to utilize up to 64GB of memory and 32 processors. Improvements in symmetric multiprocessing (SMP) support provide more operations in parallel, taking advantage of 2-32 processor systems. The popular DBCC utility now supports parallel threads, offering performance improvements equivalent to the number of system processors. Parallel index creation is also enabled, providing significant performance improvements in frequently updated transactional databases.

An improvement in server communications performance, Virtual Interface System Area Networks (VI SANs) offer ultra-high, speed server-to-server communications on dedicated, hardware-controlled connections. VI SANs are supported in SQL Server 2000 through direct relationships with Giganet (www.giganet.com) and Compaq (www.servernet.com), which offer two of the leading SAN solutions. This latest release of SQL Server offers thorough support for scale-up hardware and software configurations.

A discussion on scalability isn't complete without discussing SQL Server's support for desktop and handheld devices. SQL Server 2000 not only delivers versions designed for desktop and notebook use; it also includes SQL Server 2000 Windows CE Edition. This small-footprint SQL Server solution provides a consistent programming model that allows SQL developers to take advantage of existing skills to deliver Windows CE solutions. CE Edition replication features allow for bi-directional merge replication with central database servers through Internet Information Services. SQL Server 2000 provides a powerful platform for delivering applications that can run on any Windows device, from pocket PCs running Windows CE to 32-way servers running Windows 2000 Datacenter Server.

Distributed Partitioned Views

One of SQL Server 2000's more publicized new features is its support for distributed partitioned views. By horizontally partitioning SQL Server data and distributing it across multiple servers, called federated database servers, distributed partitioned views share the processing load across autonomous servers. This scale-out approach to scalability allows each database server to not only share the processing load, but also to access and deliver the entire table contents to client requests through linked server connections. Distributed partitioned views are updateable in SQL Server 2000. Microsoft used updateable, distributed partitioned view technology to achieve the record-breaking TPC-C benchmarks we discussed early in this chapter.

This technique of database partitioning and load distribution utilizes a shared-nothing clustering approach, which is considered superior to shared resource clustering. Shared-nothing clustering eliminates any single point of failure in the cluster and reduces bottlenecks that can occur with shared resources. Distributed partitioned views provide a significant improvement in SQL Server's scale-out capabilities for load distribution.

Fail-Over Clustering

Every application depends on the availability of its components to respond to application requests in a timely manner. Without near 100% availability, application reliability and user satisfaction will affect the success of your solution and your business. SQL Server continues to offer fail-over clustering support to provide the highest level of availability possible from your solutions. SQL Server 2000 extends and enhances fail-over clustering support with easier setup and management of SQL clusters. SQL Server fail-over clustering is dependent on Microsoft Clustering Services (MSCS), which is automatically detected during SQL Server setup, eliminating the Failover Cluster Wizard from version 7.0. Additional enhancements allow you to add or remove nodes during setup and reinstall or rebuild cluster nodes using the Enterprise Manager without affecting other cluster nodes.

SQL Server 2000 supports both active/active and active/passive clustering modes. In active/passive clustering, a dedicated, secondary server is configured and idle until fail-over is required and the secondary server takes control of the shared disks and begins processing database requests. Active/active mode clustering allows each server to operate independently with separate databases and fail-over to any remaining active nodes. Running on Windows 2000 Datacenter Server, SQL Server 2000 supports up to four-node fail-over clustering. Clustering enhancements now allow fail-over and fail-back from or to any node in the SQL Server 2000 cluster as well as support for applications or features that are server name dependent, such as replication and distributed partitioned views.

Log Shipping

SQL Server 2000 now offers integrated log-shipping capabilities. Configuring log shipping creates a "warm" standby server by continually backing up and restoring transaction logs from one database to another. These standby servers can be used in the event of a server failure or for read-only applications such as reporting, in which activity can be offloaded, eliminating the impact on the primary server. This functionality was introduced in SQL Server 7.0, but its poor documentation and configuration techniques did not allow many organizations to benefit from it. The Database Maintenance Plan Wizard in SQL Server 2000 can now be used to configure log shipping between servers.

Data Transformation Services

Data Transformation Services (DTS) was introduced with SQL Server 7.0 and has become a tremendously popular feature of SQL Server solution developers. If you haven't implemented DTS in your solutions, you should take a thorough look at DTS and its capabilities for simplifying or extending your solutions. As the second release of DTS, SQL Server 2000 includes several feature enhancements and usability additions. DTS now supports custom tasks, allowing you to develop your own DTS tasks or use third-party tasks in DTS. Based on COM, custom tasks can be built using Microsoft Visual Basic or C++. Several new tasks have also been added to DTS:

- **FTP Task** for transferring files to or from remote FTP sites or directories to a destination directory.
- **Execute Package Task** for embedded package execution.
- **Message Queue Task** for sending and receiving application messages using Microsoft Message Queue.
- **Dynamic Properties Task** for modifying runtime package properties from .INI files, query results, or variables.
- **Analysis Services Processing Task** for updating Analysis Services models.
- **Data Mining Prediction Query Task** for generating predictive results from data-mining models.
- **Transfer Tasks** for transferring jobs, logins, master stored procedures, and databases between servers.

For advanced control of data transformations, the new multiphase data pump can be used to control transformation activity and handle errors that occur during processing. The addition of parameterized queries and global variables allows developers to build dependent tasks and transfer data or status information between tasks and other packages. New advanced logging capabilities are possible, allowing stronger monitoring and tracking of DTS package activity. All of this functionality can create DTS solutions with multipackage processes, advanced error handling, and restart capabilities.

A new programmability enhancement in DTS is the support for saving DTS packages as Visual Basic source code files. These files can be included in Visual Basic projects, used as DTS programming training tools, or backed up for code and version control. This latest release of DTS addresses many of the shortcomings of the first version and offers a powerful programming environment for transferring and manipulating SQL Server data.

Replication Services

Replication features in SQL Server provide availability and performance improvements for applications in many organizations by delivering data copies and updates to multiple servers and remote, and sometimes disconnected, users. Multilocation organizations are able to keep local copies of data for reads and writes, eliminating the need for high-speed, dedicated connections to remote database servers. Mobile application users are able to access and edit data when they are disconnected from any network, allowing them to update the central database when they are able to reconnect to their corporate network or the Internet. In a growing global and mobile user environment that requires immediate and reliable access to information, replication rises to the task.

SQL Server 2000 continues to build on the successful replication features of versions 6.5 and 7.0 with numerous enhancements and additions in functionality and usability. Administering replication now offers a root-level Replication folder in Enterprise Manager for displaying all publications and subscriptions for all databases on the server. Replication Monitoring now supports viewing multiple distribution servers for centralized control and monitoring of replication across the enterprise. The Create Publication and Subscription Wizards have been enhanced to provide optional display of advanced setup options, making it easier for new users to set up replication publications and subscriptions without the burden of complex configuration options. Replication has also been enhanced to support new SQL Server 2000 features that include multiple-instance support, indexed views, new data types, and Active Directory integration.

Additional features such as schema update replication and on-demand script execution extend the capabilities of replication, allowing database changes, such as added or dropped columns, and scriptable actions to be performed on subscribers. Published Data Transformations is a new option for snapshot and transactional replication. Through integration with DTS, transformable subscriptions can modify and deliver data to subscribers based on individual requirements supporting column mapping, string manipulation, and functions as data. Additionally, transactional and merge replication now offer data validation to ensure proper replication activity.

Merge replication has been extended to offer additional conflict resolvers and support for identity and timestamp columns in publications. Several new conflict resolvers offer immediate and interactive resolution of merge replication conflicts. Management of identity columns in updating subscriptions maintains database integrity by validating identity values and enforcing primary key constraints. SQL Server 2000 also supports timestamp columns in published tables.

Availability enhancements in transactional replication include advanced error handling, backup and restore enhancements, and queued updating subscribers. Transactional replication can incur and continue under specified error conditions. Restoring a database configured for transactional replication is possible without disabling and reconfiguring replication or reinitializing publications. For “warm” standby server configuration,

transactional replication also supports log shipping, offering fail-over capabilities without reconfiguring replication.

Active Directory Integration

Making SQL Server publications available throughout the enterprise is the goal of replication's Active Directory integration. Publications in SQL Server 2000 can be published in AD using the Create Publication Wizard, Publication Properties dialog box, or T-SQL, allowing users to locate and subscribe to publications. Windows 2000's Active Directory Search tool includes a SQL Server Publications search option that allows users to search any level of the directory for publications and subscribe to them if permissions allow.

Queued Updating Subscribers

An exciting new capability of replication for remote application designers is support for queued updating subscribers in SQL Server 2000. This enhancement to updating subscriber functionality allows users who are disconnected from a network connection to the publishing server to update and edit replicated data. Once the remote user has reconnected to the publishing server, updates are propagated and the databases are resynchronized. Thorough conflict resolver options handle situations in which queued updates affect data that has been updated on the server during the period of time the remote database was disconnected. This type of functionality opens new doors for applications, such as sales force solutions, in which users are frequently traveling and high-speed connections to corporate servers are not always available. Support for Windows CE applications, running SQL Server 2000 Windows CE Edition, to participate in replication also offers new avenues of solution development on mobile devices.

Multiple Server Instance Support

A significant feature enhancement offering application isolation and greater security is SQL Server 2000's new multiple-server instance support. Organizations such as ASPs and Web-hosting providers can use multiple-instance support to offer dedicated SQL Servers to client applications while leveraging expensive hardware investments in large servers. Multiple-instance support also offers advantages to development organizations that have projects requiring specific SQL Server configurations. Multiple SQL Server instances are isolated, so configuration information is specific to each instance and application errors that could crash a specific instance of SQL Server will not affect the remaining instances.

During the SQL Server setup process, you can elect to install an additional named instance of SQL Server, which offers side-by-side execution of multiple SQL Servers running on the same computer. Multiple-instance support allows a single instance of SQL Server 6.5 or 7.0 to exist along with one or more instances of SQL Server 2000. A maximum of 16 instances are supported by SQL Server 2000.

A default instance of SQL Server 2000 can be installed and works like previous versions, allowing applications to connect to SQL Server using just the computer name. Each SQL Server 2000 named instance can be accessed by adding the instance name to the machine name, such as SQLSERVER1\INSTANCE1 for accessing the named instance INSTANCE1 installed on the server named SQLSERVER1.

Each SQL Server instances includes separate copies of:

- User and system databases
- SQL Server and SQL Server Agent Services; instance services are named MSSQL$ *instancename* and SQLAgent$ *instancename*
- Registry keys for the database engine, SQL Server, and SQL Agent services
- Network connection address for accessing individual SQL Server instances

The following components are shared between instances of SQL Server:

- One SQL Server program group and utilities are installed from the first SQL Server 2000 instance. If you install a SQL Server 2000 instance on a computer running a previous version of SQL Server, SQL tools, such as the Query Analyzer, are upgraded, and previous versions will no longer be accessible. Most of the functionality from previous versions has been carried over to SQL Server 2000, and backward compatibility will allow you to use Enterprise Manager and Query Analyzer on previous versions of SQL Server. You will, however, still be able to access the Books Online (BOL) from previous SQL versions.
- One instance of Microsoft Search (MSSearchService) exists for processing full-text search requests against all SQL Server instances.
- One instance of Microsoft English Query is installed.
- One instance of Microsoft Analysis Server is installed for OLAP and data-mining capabilities.
- One set of registry keys for client applications is configured.
- One set of development libraries and samples is installed.

Web and Internet Standards Support

Microsoft's vision of the future through its .NET architecture includes the integration of industry-standard protocols and technologies as well as enabling Web access to products and solutions. SQL Server 2000 is the first release to include extensive, integrated support for XML, a core component of .NET and an increasingly used data communications method between heterogeneous and Internet-connected information systems. In addition to XML integration, SQL Server 2000 offers Web access and secure network communications by providing support for the HTTP and SSL protocols.

Web Access Using Hypertext Transfer Protocol

Through integration with Microsoft Internet Information Services (IIS), users can query and update SQL Server databases using direct SQL, template queries, and XPath queries

over HTTP and HTTPS. With Web access to SQL Server databases, developers can begin rapidly delivering Web services and enabling their applications for communication with SQL Server using HTTP. Using the Configure SQL XML Support in IIS utility, specific databases can be configured for access through IIS, allowing users to issue queries over HTTP. The following example selects all products from the Northwind database's Products table and returns the results in XML format using the FOR XML statement clause. This example uses a virtual directory, created on the Web server named SERVER1, labeled NORTHWIND, which has been mapped to the Northwind sample database:

```
http://server1/northwind?sql=SELECT+*+FROM+PRODUCTS+FOR+XML+AUTO&root=
root
```

Web access in SQL Server 2000 extends to accessing Analysis Services using HTTP and HTTPS. IIS can be configured for secure access to OLAP cubes using Web protocols, allowing remote applications and services to analyze and link to existing cubes. This capability allows organizations to deliver analysis services to remote locations and as a potential service to clients.

Secure Sockets Layer

Secure Sockets Layer (SSL) has become an industry standard for securing and validating network communications around the world. Most organizations use SSL to secure Web site activity such as e-commerce over the Internet. Aside from using SSL to encrypt and validate Web access to SQL Server using IIS and HTTPS, Microsoft now supports SSL for encrypting data communications directly with SQL Server. By configuring the SQL Server computer with a valid SSL certificate, application computers can securely exchange information with SQL Server.

Note SQL Server 2000's support SSL requires a valid SSL certificate installed on the SQL Server computer. SSL encryption is currently available in either 40-bit or 128-bit encryption, offering different levels of data protection. With any encryption process comes a loss in performance because data needs to be encrypted at the source and then unencrypted at the destination. Using SSL access from connecting computers requires that the connecting computer have a root certificate authority (CA) from the issuer of the SSL certificate. This allows the connecting computer to authenticate the certificate as valid and use it for secure communications.

Support for SSL reflects Microsoft's approach to using industry-standard protocols rather than previously proprietary implementations. Multiprotocol encryption, which uses the Windows RPC encryption application programming interface (API), continues to be supported in SQL Server 2000.

SQL Server 2000 Versions, Features, and Requirements

To meet Microsoft's goal of delivering software on any device and maintaining the integral role of Microsoft SQL Server to its application architecture has resulted in a somewhat confusing range of SQL Server 2000 product editions. SQL Server 2000 is available in the following editions:

- SQL Server 2000 Enterprise Edition
- SQL Server 2000 Standard Edition
- SQL Server 2000 Personal Edition
- SQL Server 2000 Desktop Engine (MSDE)
- SQL Server 2000 Developer Edition
- SQL Server 2000 Windows CE Edition

Platform support for SQL Server ranges from Windows CE to Windows 2000 Datacenter Server. The requirements and features available on each of these platforms are different across each SQL Server edition. Under-standing the hardware requirements, operating system compatibility, feature differences, and licensing rules is important to selecting and implementing SQL Server 2000 in your enterprise.

The following sections review each of these SQL Server editions to give you a clear understanding of which edition is right for your application and enterprise needs. The chart in Table 1.2 provides a comprehensive comparison of the editions' feature sets.

Common Edition Requirements

With the exception of SQL Server 2000 Windows CE Edition, the following common system requirements apply to all SQL Server 2000 editions.

Processor architecture requirements

- Intel Pentium or compatible 166MHz processor.

Memory (RAM)

- Minimum of 64MB.

Disk space (SQL Server and related components)

- Depending on installed components, 95–270MB. A typical installation requires 250MB of hard disk space. SQL Server 2000 Desktop Engine requires 44MB of hard disk space.
- Analysis Services: Minimum installation requires 50MB, typical installation requires 130MB.
- Microsoft English Query: Minimum installation requires 80MB.

Networking

- Transmission Control Protocol/Internet Protocol (TCP/IP) must be enabled before installing SQL Server 2000. TCP/IP is the default Net-Library for SQL Server 2000.

Additional requirements

- Microsoft Internet Explorer 5.0 or higher is required. Microsoft Management Console (MMC)/Enterprise Manager and HTML help are dependent on common components installed with IE 5.0 or higher.
- Web Access to SQL Server and/or Analysis Services requires Microsoft Internet Information Server/Services 4.0 or higher.
- Client connectivity-only setup option installs ActiveX Data Objects 2.6 for accessing SQL Server 2000 databases. ActiveX Data Objects 2.6 requires IE 4.01 SP2 or higher.
- Connectivity-only option requires IE 4.01 sp2.

For specific hardware compatibility, refer to the hardware compatibility list (HCL) from Microsoft (www.microsoft.com/hcl) for the SQL Server 2000-compatible operating system being used.

SQL Server Licensing and Pricing

Of the seven different versions of SQL Server 2000, only three versions can be purchased: Enterprise Edition, Standard Edition, and Developer Edition. With SQL Server 2000, Microsoft has introduced its new processor-based licensing model option in addition to offering traditional server/client access licensing (CAL):

- **Processor Licensing Model** Each processor installed in the machine running SQL Server requires a license. Unlimited client access is included with processor licensing, eliminating the need to purchase individual CALs for client connectivity. Processor licensing replaces the Internet Connector License required by previous versions to support Web-enabled database solutions such as e-commerce.
- **Server/Client Access Licensing Model** Each server running SQL Server requires a server license, and each client accessing the SQL Server requires a CAL. Server/CAL licensing is an effective solution for organizations deploying SQL Server solutions in limited user environments.

Table 1.1 represents the estimated retail pricing under each licensing model for SQL Server 2000 full and upgrade versions, which include a specified number of CALs.

Table 1.1: Microsoft SQL Server 2000 License Estimated Retail Pricing

Product	Processor Licensing	Server/Client Access Licensing (CAL)
Enterprise Edition	$19,999 per processor	$11,099 with 25 CALs $5,049 with 25 CALs (Upgrade Version)
Standard Edition	$4,999 per processor	$1,489 with 5 CALs $749 with 5 CALs (Upgrade Version) $2,249 with 10 CALs
Developer Edition	N/A	$549 with 8 CALs

Upgrade pricing is available for organizations upgrading from previous versions of Microsoft SQL Server as well as competitive upgrades from the following RDBMSs:

- Oracle 7.x and 8.x
- IBM DB2 6.x
- Informix 7.x
- Sybase system 1.x
- Progress 8.x and 9.x

Personal Edition, Desktop Engine, and CE Edition licensing is included with specific server versions shown in Table 1.1 . Review the following sections for information on licensing and availability of these editions. For additional information such as volume discounts and updates on SQL Server licensing and pricing, visit the Microsoft SQL Server Web site at www.microsoft.com/sql .

Enterprise Edition

SQL Server 2000 Enterprise Edition is the flagship version of SQL Server and is the only version that supports all the features in SQL Server 2000, including high availability and multiserver options such as fail-over clustering, log shipping, distributed partitioned views, and indexed views. The capability, performance, and availability features of Enterprise Edition are designed for OLTP and data warehousing applications such as e-commerce and decision support systems.

Hardware Requirements and Capacity Limits

SQL Server 2000 Enterprise Edition supports the following hardware configurations:

Processor architecture requirements

- Intel Pentium or compatible 166MHz processor
- SMP support:
    - One to 32 processors with Windows 2000 Datacenter Server

    - One to eight processors with Windows 2000 Advanced Server and Windows NT 4.0 Server, Enterprise Edition
    - One to four processors with Windows 2000 Server and Windows NT 4.0 Server

Memory (RAM)

- 64MB minimum; a minimum 128MB of memory is recommended for running SQL Server 2000 Enterprise Edition on Windows 2000 Server
- Maximum memory supported:
    - 64GB with Windows 2000 Datacenter Server
    - 8GB with Windows 2000 Advanced Server
    - 4GB with Windows 2000 Server
    - 3GB with Windows NT 4.0 Server, Enterprise Edition
    - 2GB with Windows NT 4.0 Server

Disk space (MB for SQL Server and related components)

- 95–270MB for SQL Server Enterprise Edition, depending on installed components; a typical installation requires 250MB of hard disk space

Operating System Compatibility

SQL Server 2000 Enterprise Edition is compatible with the following operating system versions:

- Microsoft Windows 2000 Datacenter Server
- Microsoft Windows 2000 Advanced Server
- Microsoft Windows 2000 Server
- Microsoft Windows NT 4.0 Server Enterprise Edition (SP5 or higher)
- Microsoft Windows NT 4.0 Server (SP5 or higher)

Standard Edition

SQL Server 2000 Standard Edition, the base SQL Server product, is recommended for departmental and small application deployments. The Standard Edition of SQL Server does *not* support availability enhancements such as fail-over clustering and log shipping as well as performance enhancements, including distributed partitioned views, indexed views, and parallel index operations. Advanced Analysis Services features such as HTTP access and linked cubes are not available in SQL Server 2000 Standard Edition. SQL Server Standard Edition offers a subset of the capabilities provided by Enterprise Edition and offers no features that are not available in Enterprise Edition.

Hardware Requirements

SQL Server 2000 Standard Edition supports the following hardware configurations:

Processor architecture requirements

- Intel Pentium or compatible 166MHz processor
- SMP support:
    - One to four processors with Windows 2000 Server, all editions
    - One to eight processors with Windows NT 4.0 Server, Enterprise Edition
    - One to four processors with Windows NT 4.0 Server

Memory (RAM)

- 64MB minimum
- Maximum memory supported:
    - 2GB with Windows 2000 Server, all editions
    - 2GB with Windows NT 4.0 Server, all editions

Disk space (MB for SQL Server and related components)

- 95–270MB for SQL Server Enterprise Edition, depending on installed components; a typical installation requires 250MB of hard disk space

Software and Operating System Compatibility

SQL Server 2000 Standard Edition is compatible with the following operating system versions:

- Microsoft Windows 2000 Datacenter Server
- Microsoft Windows 2000 Advanced Server
- Microsoft Windows 2000 Server
- Microsoft Windows NT 4.0 Server Enterprise Edition (SP5 or higher)
- Microsoft Windows NT 4.0 Server (SP5 or higher)

Personal Edition

SQL Server 2000 Personal Edition replaces SQL Server 7.0 Desktop Edition. Personal Edition is included with SQL Server 2000 Enterprise or Standard Edition processor and CAL licensing and cannot be purchased as a separate product. Personal Edition provides nearly all the features of Standard Edition but has major limitations in scalability. The workload governor in Personal Edition has been tuned for five concurrent users, limiting its capabilities in multiuser environments. In addition, transactional replication in Personal Edition is supported only as a subscriber. Full-Text Search and Analysis Services are not available when Personal Edition is installed on a Windows 98 computer.

Personal Edition offers support for Windows 2000 Professional, Windows NT 4.0 Workstation, Windows 98, and Windows Millennium Edition (ME), offering local application data storage and access to the SQL Server graphical user interface (GUI) management tools.

Hardware Requirements

SQL Server 2000 Personal Edition supports the following hardware configurations:

Processor architecture requirements

- Intel Pentium or compatible 166MHz processor
- SMP support:
    - One to four processors with Windows 2000 Server, all editions
    - One to eight processors with Windows NT 4.0 Server, Enterprise Edition
    - One to four processors with Windows NT 4.0 Server
    - One or two processors with Windows 2000 Professional, Windows NT 4.0 Workstation
    - One processor with Windows 98, Windows Millennium Edition

Memory (RAM)

- 64MB minimum
- Maximum memory supported:
    - 2GB with Windows 2000 Server, all editions
    - 2GB with Windows NT 4.0 Server, all editions

Disk space (MB for SQL Server and related components)

- 95–270MB for SQL Server Enterprise Edition, depending on installed components; a typical installation requires 250MB of hard disk space

Software and Operating System Compatibility

SQL Server 2000 Standard Edition is compatible with the following operating system versions:

- Microsoft Windows 2000 Datacenter Server
- Microsoft Windows 2000 Advanced Server
- Microsoft Windows 2000 Server
- Microsoft Windows 2000 Professional
- Microsoft Windows NT 4.0 Server Enterprise Edition (SP5 or higher)
- Microsoft Windows NT 4.0 Server (SP5 or higher)
- Microsoft Windows NT 4.0 Workstation (SP5 or higher)
- Microsoft Windows 98 and Millennium Edition

Developer Edition

SQL Server 2000 Developer Edition offers the complete functionality of Enterprise Edition but has a restricted licensing model. Developer Edition is designed for development and test installations only and cannot be licensed for production

environments. Complete Enterprise Edition functionality and hardware compatibility are available, allowing developers to build solutions that take advantage of the full range of SQL Server features. In addition to supporting the full range of operating systems supported by Enterprise Edition, Developer Edition can be installed on Windows 2000 Professional and Windows NT 4.0 Workstation with support for up to two processors.

SQL Server 2000 Desktop Engine

Microsoft introduced the Microsoft Data Engine (MSDE) for distributing applications requiring a robust local data storage system. This common data storage and management engine offers scalability, security, and compatibility with both Microsoft Access 2000 and SQL Server 7.0. SQL Server 2000 includes an updated and renamed version of MSDE, Microsoft Desk-top Engine (MSDE). MSDE provides the SQL Server 2000 database engine as well as support for replication; transactional replication is restricted to subscriber mode only. MSDE offers Personal Edition features with the exception of the graphical administration tools and additional features such as Analysis Services and Full-Text Search capabilities. MSDE is designed as a distribution method for SQL Server- or Microsoft Access 2000-based solutions such as those built using Microsoft Office 2000 or Microsoft Visual Studio. There are no CAL requirements for the distribution of MSDE as long as MSDE connects to no SQL Server 2000 Enterprise or Standard Edition servers. Applications using MSDE that connect to SQL Servers require processor or CAL licensing.

Deployment of MSDE has been made considerably easier by the addition of numerous flags that help with setting the targetdir, datadir, upgrade mode, security settings, and so on. Previously, all this information had to be gathered from and written to an *.iss file.

SQL Server 2000 Windows CE Edition

An exciting addition to the SQL Server family, SQL Server 2000 Windows CE Edition was released in October 2000. This edition offers the fundamental SQL Server 2000 features and compatibility in a small footprint designed for Windows CE devices. Programming compatibility with personal and server editions of SQL Server 2000 provides SQL developers an easy transition to building CE-based applications. Support for merge replication allows CE-based applications to participate in read-only and read/write database applications with centralized SQL Servers.

The software components of SQL Server Windows CE Edition can be installed only on Windows CE devices. SQL Server 2000 Window CE Edition is included with Developer Edition licensing and may be freely distributed with applications that do not connect to SQL Servers. SQL Server CE Edition applications that connect to a central SQL Server 2000 Server require processor licensing. CE applications that connect to previous versions of SQL Server require CALs. This new SQL Server-compatible database engine will open new doors to handheld application development, finally allowing Windows CE devices to play a strong role in corporate application solutions by providing a familiar and strong data management architecture.

Table 1.2 provides a comparison of all the SQL Server editions.

Table 1.2: SQL Server 2000 Edition Features Comparison

Feature	Enterprise Edition	Standard Edition	Personal Edition	Developer Edition	Desktop Engine	CE Edition
Multiple-instance support	X	X	X	X	X	
Fail-over clustering	X			X		
Log shipping	X			X		
Parallel DBCC	X			X		
Parallel CREATE INDEX	X			X		
Enhanced read-ahead and scan	X			X		
Indexed views	X			X		
Federated database servers	X			X		
System area network (SAN)	X			X		
Graphical administrative tools	X	X	X	X		
Graphical utilities for language	X					
Full-Text Search	X	X	X (except Windows 98)	X		
Snapshot replication	X	X	X	X	X	
Transactional replication	X	X	Subscriber only	X	Subscriber only	
Merge	X	X	X	X	X	Anonymous

Table 1.2: SQL Server 2000 Edition Features Comparison

Feature	Enterprise Edition	Standard Edition	Personal Edition	Developer Edition	Desktop Engine	CE Edition
replication						subscriber only
Immediate updating subscribers	X	X	X	X	X	
Queued updating subscribers	X	X	X	X	X	
Analysis Services	X	X	X	X		
User-defined OLAP partitions	X			X		
Partition Wizard	X			X		
Linked OLAP cubes	X			X		
ROLAP dimension support	X			X		
HTTP Internet support	X			X		
Custom rollups	X	X	X	X		
Calculated cells	X			X		
Write-back to dimensions	X			X		
Very large dimension support	X			X		
Actions	X	X	X	X		
Real-time OLAP	X			X		
Distributed partitioned	X			X		

Table 1.2: SQL Server 2000 Edition Features Comparison

Feature	Enterprise Edition	Standard Edition	Personal Edition	Developer Edition	Desktop Engine	CE Edition
cubes						
Data mining	X	X	X	X		
English Query	X	X	X	X		

Should You Migrate to SQL Server 2000?

Microsoft SQL Server 2000 offers the best performance, scalability, and reliability of any SQL Server release. Building on the success of SQL Server 7.0, Microsoft inflicted nearly double the amount of functional and stress testing against SQL Server 2000 as version 7.0, making it the strongest release yet. Microsoft uses SQL Server to manage its own business operations, and before public release, the company successfully migrated 40 of the internal business systems to SQL Server 2000. Statistics such as these are interesting, but I'm sure many of you are thinking: Of course it would be easy to migrate to SQL Server 2000 when the product developers are on your staff.

Successful implementation of SQL Server 2000 does not end at Microsoft. Numerous organizations have recorded record performance measures, including SAP R/3 tests that resulted in 53 percent greater scalability over Oracle, supporting 7500 concurrent users. Previous performance records held by Sun/Oracle on the J.D. Edwards OneWorld Benchmark were beaten by 37 percent, providing support for over 3442 concurrent users. Organizations such as Dell Computer Corporation enjoy 99.99 percent availability on their SQL Server-based Web applications. Companies such as MasterCard, Nestlè, Chase Manhattan Bank, Ford, Honeywell, and many others implement SQL Server to support their operations, with tremendous success. All these situations should answer the question of whether SQL Server 2000 is capable of supporting the scalability and reliability demands of your applications.

Note Based on Transaction Processing Council test results (TPC-C, October 2000), SQL Server 2000 provides the highest levels of performance on Windows 2000, at the lowest per-transaction cost of any competing database system, including those from Oracle and IBM. SQL Server 2000 completed 505,302 transactions at a cost of $20.68, compared with IBM at 440,879 transactions at a cost of $32.28 and Oracle at 144,331 transactions at a cost of $52.75. SQL Server's climb to the top of the TPC benchmarks is the result of scalability and performance enhancements in SQL Server 2000.

The TPC benchmarks are considered industry-standard database performance measures. For more information on TPC and the benchmark results as well as updates to these findings, visit www.tpc.org.

How Will SQL Server 2000 Benefit My Organization?

SQL Server 2000 includes many new and enhanced features that have proven beneficial to all types of organizations and applications, including e-commerce, business intelligence, and line-of-business applications. Integrated technologies such as XML support, OLAP, and data-mining engines offer an unprecedented list of features, allowing SQL Server to play an integral role in every aspect of your organization—from business-to-business integration and electronic commerce to back-office data analysis and decision support. The importance of technologies such as XML continues to increase as organizations work toward greater integration with business partners, providing higher

levels of efficiency and access to new customers. OLAP and data-mining capabilities result in more successful business decisions based on the discovery of new information among your piles of data.

Whether your organization is a small business or a multinational corporation, SQL Server 2000 offers advantages such as improved self-tuning, automatic file growth, and configuration wizards through four-node fail-over clustering, federated servers, and support for up to 32 processors and 64GB of memory. SQL Server 2000 offers compatible platform support ranging from Windows CE to Windows 2000 Datacenter Server, allowing organizations to leverage existing SQL programming skills to deliver applications on every Windows platform.

Large and mission-critical applications can take advantage of SQL Server 2000's scalability enhancements, providing thorough support for scale-out and scale-up practices. Database partitioning and federated servers allow organizations to take advantage of low-cost, commodity hardware. SQL Server 2000's support for 64GB of memory and 32-way SMP systems will deliver the performance available on high-end, mainframe-class computers from vendors such as Unisys and Hewlett-Packard.

Will SQL Server 2000 Fit into My Organization?

Whether you are currently using SQL Server or have implemented a different RDBMS, SQL Server 2000 provides the tools for a successful migration or peaceful and productive coexistence. For organizations upgrading to SQL Server 2000 from SQL Server 6.5 and 7.0, backward compatibility and upgrade wizards make the migration seamless and efficient, with no modifications to your existing applications in most situations. A SQL Server 2000 named instance installation can even run concurrently with SQL Server 6.5 or 7.0 on the same server, so your database server can remain active during your migration. Tools such as the new Copy Database Wizard minimize database server downtime by copying databases and configuration information from SQL Server 7.0 servers without affecting their ability to service active applications.

Migrating from other RDBMSs or integrating SQL Server 2000 into heterogeneous environments is easily accomplished with its support for OLE DB providers for Oracle and Sybase, to name a couple. Nearly all popular database and information storage providers offer OLE DB drivers, allowing SQL Server to connect to other systems and perform tasks such as data import or export, execute distributed queries, and work with remote data sets using the OPENROWSET function. DTS in SQL Server allow you to attach to remote database systems and migrate objects and data. All SQL Server 2000 integration and migration capabilities will assist your organization in making a smooth transition to SQL Server 2000.

Steps to a Successful SQL Server Migration

The task of migrating existing applications and infrastructure to a new platform can bring on cold sweats for many database and system administrators, but with proper planning and execution, migrating to SQL Server 2000 can be a peaceful and rewarding experience. The first step to delivering a successful migration is to consider the impact of this release of SQL Server on your organization—from administration and training requirements to potential application capability enhancements. After you have evaluated the business impacts of SQL Server 2000 and decided to take advantage of its performance, reliability, and capabilities, your migration should include the following steps:

1. **Determine the scope of your SQL Server 2000 migration.** Before you can begin your migration, you must determine who or what will be affected by your migration. These factors include computers, applications, users, and technology personnel.
2. **Inventory systems that will be involved in the migration.** Perform a comprehensive analysis of existing server configurations, including hardware and software.
3. **Train administrators and support personnel on SQL Server 2000.** Organize and develop a training program for the technology professionals who will implement and support the migration and production environment. Nothing is more critical to the success of your migration than knowledgeable administrators and support personnel.
4. **Develop the migration and test plan.** Design and document your migration strategy, including scheduling, implementation and rollback procedures, test lab configuration, and testing procedures for both the database servers and affected applications.
5. **Perform a test migration.** Implement your migration strategy in a test lab environment that closely mirrors your production environment. Although the costs sometimes make this difficult, being familiar with the installation and configuration process will be priceless when it comes to implementing your migration in production.
6. **Implement production environment migration.** Roll out your tested migration plan to your production servers.
7. **Test the production migration.** Review and test your production servers immediately following your completed migration. All premigration features should be revalidated for availability and performance.
8. **Monitor and optimize the production environment.** Continue monitoring your new production environment to discover optimal configuration settings.

Planning a SQL Server Migration

The success of every project depends on a thorough project plan that includes analysis, implementation, testing, and monitoring. It is important that you consider all the factors that encompass your choice to migrate to SQL Server 2000, along with the benefits and

potential problems that migrating might create for your applications, users, and support personnel. SQL Server 2000's performance, reliability, and scalability enhancements are reason enough for many organizations to choose SQL Server 2000. Understanding your applications and how they can take advantage of specific features of SQL Server 2000 will mean the difference between simply upgrading your database server and improving your applications.

Once you have decided to either upgrade or migrate to SQL Server 2000, the work begins. Every migration plan should include a complete inventory of your existing infrastructure and hardware. Simple and obvious questions will arise. If your applications can benefit from SQL Server 2000's Active Directory integration, has AD been implemented in your organization? Can you migrate to SQL Server 2000 now and add this advanced functionality when AD becomes available? Can your applications take advantage of SQL Server 2000's database partitioning, federated servers, and distributed partitioned views for greater scalability? If the answer is yes, what effect does this change have on your applications? These questions can go on and on as you begin to discover exactly what SQL Server 2000 can do for you and your applications. This discovery process should be the second step to planning your migration because it will steer you in the appropriate direction to answer questions regarding issues such as which components of SQL Server 2000 you will need to install and configure and whether you need additional server or network hardware to take advantage of fail-over clustering, scalability, or administration.

Determine Existing and Future Application Requirements

Most organizations know exactly what their applications do on a daily basis. The goal of analysts and designers is to come up with exactly what those applications are *capable* of doing. Knowing the direction of your applications and your business will not only help in planning and implementing your migration, it will also help in training your personnel and administering your servers to take advantage of the features that best fit your applications and organization. Features such as Web access to SQL Server, English Query, and Analysis Services require additional planning, installation, and configuration steps. Being aware of what your database servers will do is important to capacity and configuration planning.

Inventory Existing Database Servers

Before you can determine how you will roll out SQL Server in your organization, you need a complete inventory of existing server configurations, including hardware and operating system versions. SQL Server 2000 is designed for Windows 2000 Server, and specific versions of Windows 2000 will allow you to take advantage of its scalability enhancements. Many organizations still run on Windows NT 4.0, and SQL Server 2000 is compatible with Windows NT 4.0, allowing you to leverage existing configurations. If you have existing SQL Server installations, analyzing and developing an upgrade plan will be part of your migration. If you will be migrating from other RDBMSs, additional

planning and hardware might be necessary to complete your migration. All these questions and answers depend on your existing server configurations.

Train Database Administrators and Support Personnel

Training is probably one of the most overlooked assets in many organizations. With the fast pace of technology, on-the-job training seems to be the common method of support and administrator training for some organizations. This ad hoc approach can lead to big problems in terms of system reliability and performance. Although SQL Server 2000 includes very successful self-tuning capabilities, allowing smaller organizations to take advantage of SQL Server with little administration requirements, knowing the architecture of SQL Server will be a lifesaver the first time your server decides that it's all done working!

With SQL Server's significant importance to your organization, your database administrators and application developers should be trained in SQL Server 2000 before you even begin to complete your migration plan. Allowing your administrators to understand the architecture and features of SQL Server 2000 and its programming techniques will make them efficient at monitoring and tuning application performance as well as creating databases and users. For developers, a keen understanding of SQL Server's programming techniques is critical, but understanding the architecture and administrative tasks of SQL Server 2000 will allow them to design more efficient applications. We've all seen applications created by developers who had no consideration for network bandwidth or data storage requirements.

Next Steps to Successful SQL Server Migration

Documenting your environment, analyzing your applications, training your personnel, and devising upgrade or installation procedures will give you a strong foundation for delivering a successful implementation of SQL Server 2000. Before you can begin rolling out SQL Server in your organization, it is critical that you test your migration plan. On paper or in theory, your detailed plan is flawless, but all it takes is one missed installation option or a misunderstood dialog box to render your server helpless. Make sure that your technology personnel have walked through the upgrade or migration process in a test lab where problems do not lead to business downtime and profit loss. Set up any existing applications that will be using your new database servers and test them for integration issues. All this lab work will help you deliver a successful migration, and your development teams will be able to use your lab configuration to update and test their applications as you migrate them to take advantage of SQL Server 2000.

Migrating to SQL Server 2000

As with any migration, the impact it will have on existing infrastructure and application architectures requires extra consideration because very few application environments are the same. Migrating to any version of SQL Server must include reviewing backward-compatibility issues and complex configuration environments such as installations that

have deployed replication or clustering technologies. Upgrading existing SQL Server installations that participate in replication or clustering requires additional steps to ensure a successful transition to SQL Server 2000.

If your organization is migrating from a different RDBMS, you should review in detail the system-level differences among these database systems. Every RDBMS has its own architecture and methods of storing system and configuration information as well as implementing standard database features such as SQL. Common differences such as system table names, system procedures, functions, SQL implementations, and even data types could affect your applications. By training your database administrators and SQL application developers on SQL Server 2000, the differences will become apparent, and examining your applications for possible conflicts will be easier to accomplish, allowing you to deliver a successful application migration.

SQL Server Backward Compatibility

SQL Server 2000 implements backward compatibility for nearly all the functionality included in SQL Server 6.5 and 7.0. Most SQL Server applications require no modification to run properly and enjoy the benefits of SQL Server 2000, but because SQL Server 2000 delivers an improved relational database engine, some application dependencies might need to be updated.

If you are upgrading from SQL Server 6.5 to SQL Server 2000, the following changes could affect your application:

- Several version 6.5 system tables have been eliminated from SQL Server 2000. They include master.dbo.spt_datatype_info, sysprocedures, sysbackupdetail, sysrestoredetail, sysbackuphistory, sysrestorehistory, syshistory syssegments, syskeys, systasks, syslocks, and sysusages.
- Several version 6.5 system-stored procedures have been eliminated from SQL Server 2000. If your application is dependent on any system-stored procedures, verify that they exist and produce the desired results in SQL Server 2000.
- SQL Server 2000 uses files and file groups instead of segments and devices for physical database storage. Application dependencies on segment and device storage objects no longer work.
- Task, replication, and device objects have changed, and several new system tables and stored procedures are available to support new SQL Server features.
- To allow upgraded databases to support merge and updating replication on tables that have previous table updating triggers, enable nested triggers support using the sp_configure system-stored procedure.
- The SQL Server 2000 upgrade process will retain the compatibility level specified in the original database.

If you are upgrading from SQL Server 7.0 to SQL Server 2000, the following changes could affect your application:

- Multiple-instance support in SQL Server 2000 is not supported by Multiserver Jobs in SQL Server 7.0.
- Client connectivity in SQL Server 7.0 requires an alias to access SQL Server 2000 named instances.
- SET ROWCOUNT is ignored on INSERT statements against remote tables.
- Several server configuration options are no longer supported. They include default sortorder id, resource timeout, extended memory size, spin counter, language in cache, time slice, language neutral full-text, unicode comparison style, max async IO, and unicode locale id.
- Several reserved keyword changes have been made to SQL Server 2000. No longer supported as reserved keywords are AVG, COMMITTED, CONFIRM, CONTROLROW, COUNT, ERROREXIT, FLOPPY, ISOLATION, LEVEL, MAX, MIN, MIRROREXIT, ONCE, ONLY, PERM, PERMANENT, PIPE, PREPARE, PRIVILEGES, REPEATABLE, SERIALIZABLE, SUM, TAPE, TEMP, TEMPORARY, UNCOMMITTED, and WORK. New reserved keywords include COLLATE, FUNCTION, and OPENXML.
- Changes and extensions to the DTS and SQL-DMO object models to support new functionality could affect your custom applications.
- SQL-SCM API is no longer supported.
- The SQL Server 2000 upgrade process will set the compatibility level of your database to 80, the default compatibility level for SQL Server 2000.

Tip Under certain circumstances, your applications could require Transact-SQL statement behavior compatibility with previous SQL Server versions. To support this scenario and allow these applications to run on SQL Server 2000 until they can be fully upgraded, SQL Server's database version compatibility level can be set using the sp_dbcmptlevel system-stored procedure. Valid database compatibility levels are 60 (version 6.0), 65 (version 6.5), 70 (version 7.0), and 80 (SQL Server 2000).

Upgrading from SQL Server 6.5

You *cannot* perform a direct upgrade from SQL Server 6.5 to SQL Server 2000 using the setup process. Microsoft includes the SQL Server Upgrade Wizard for migrating from SQL Server 6.5 to SQL Server 2000. The SQL Server Upgrade Wizard will prompt you through the process of migrating your SQL Server 6.5 databases and configuration to SQL Server 2000. The following information is important to upgrading from SQL Server 6.5 to SQL Server 2000:

- To upgrade from SQL Server 6.5 to SQL Server 2000 on separate computers, SQL Server 6.5 requires SP3 or higher. To upgrade from SQL Server 6.5 to a named instance of SQL Server 2000 on the same computer, SQL Server 6.5 requires SP5a or higher.
- Users will not be able to access the SQL Server 6.5 instance during the upgrade process.
- User-defined messages created using the sp_addmessage system-stored procedure will not be migrated to SQL Server 2000.

Upgrading from SQL Server 7.0

You *can* perform a direct upgrade from SQL Server 7.0 to SQL Server 2000 using the setup process. The SQL Server 2000 setup process automatically detects your existing installation of SQL Server 7.0 and prompts you to upgrade the SQL Server 7.0 installation. If you choose to upgrade your existing installation, your SQL Server 7.0 instance will be completely overwritten and will no longer be accessible. If you want both SQL Server 7.0 and 2000 to coexist and upgrade only certain databases, you can install a named instance of SQL Server 2000 on the same computer. Using the Copy Database Wizard, import the desired database into your SQL Server 2000 instance. Your SQL Server 7.0 instance will serve as the default instance on that computer, and access to your SQL Server 2000 named instance will provide access to your upgraded database.

After you have completed your SQL Server 7.0 to 2000 upgrade, you should rebuild any full-text indexes and update SQL Server statistics on your database. During the upgrade process, SQL Server 2000 setup will disable the full-text option on your database. Due to the time and resources necessary to rebuild full-text indexes, you should schedule repopulation of the full-text catalogs at a convenient time when your server load is minimal. To optimize query performance of your upgraded SQL Server database, you should update statistics for your database. You can use the sp_updatestats system-stored procedure to update statistics for all user-defined tables in your database.

Upgrading to SQL Server 2000 Fail-Over Clustering

To upgrade SQL Server 6.5 and 7.0 servers that participate in SQL Server clustering, you must first use the Cluster Wizard to uncluster the SQL Server 6.5 or 7.0 installations. After you have unclustered the previous installation, the installation process depends on the clustering mode of your previous installation. The process is similar for both versions in active/passive mode and in active/active mode. The following sections provide you with an overview of the upgrade process for fail-over clustered servers.

Upgrading from SQL Server 6.5: Active/Passive Mode

1. Uncluster your SQL Server 6.5 installation using the Cluster Wizard utility.
2. Install a default instance of SQL Server 2000.
3. Upgrade your SQL Server 6.5 installation using the SQL Server Upgrade Wizard.
4. Uninstall SQL Server 6.5.
5. Using the SQL Server 2000 setup utility, upgrade your SQL Server 2000 installation to a clustered instance.

Upgrading from SQL Server 6.5: Active/Active Mode

1. Uncluster your SQL Server 6.5 installation on Nodes 1 and 2 using the Cluster Wizard utility.
2. Install a default instance of SQL Server 2000 on Node 1.

3. Upgrade your SQL Server 6.5 installation using the SQL Server Upgrade Wizard on Node 1.
4. Uninstall SQL Server 6.5 on Node 1.
5. Install a clustered named instance of SQL Server 2000 on Node 1.
6. Use the Copy Database Wizard to migrate all databases and configuration information from the SQL Server 2000 default instance to the SQL Server 2000 clustered instance on Node 1.
7. Uninstall the SQL Server 2000 default instance on Node 1.
8. Install a default instance of SQL Server 2000 on Node 2.
9. Upgrade your SQL Server 6.5 installation on Node 2 using the SQL Server Upgrade Wizard.
10. Uninstall SQL Server 6.5 on Node 2.
11. Using the SQL Server 2000 setup utility, upgrade your SQL Server 2000 default instance on Node 2 to a clustered instance.

Upgrading from SQL Server 7.0: Active/Passive Mode

1. Uncluster your SQL Server 7.0 installation on Node 1 using the Cluster Wizard utility and reboot the Node 1 server.
2. Using the SQL Server 2000 setup utility, upgrade the SQL Server 7.0 default instance installation on Node 1.
3. Using the SQL Server 2000 setup utility, upgrade the SQL Server 2000 default instance to a clustered instance.

Upgrading from SQL Server 7.0: Active/Active Mode

1. Uncluster your SQL Server 7.0 installation on Node 1 using the Cluster Wizard utility and reboot the Node 1 server.
2. Uncluster your SQL Server 7.0 installation on Node 2 using the Cluster Wizard utility and reboot the Node 2 server.
3. Install a clustered named instance of SQL Server 2000 on Node 1.
4. Use the Copy Database Wizard to migrate all databases and configuration information from the SQL Server 7.0 instance to the SQL Server 2000 instance on Node 1.
5. Uninstall the SQL Server 7.0 instance on Node 1.
6. Using the SQL Server 2000 setup utility, upgrade the SQL Server 7.0 default instance installation on Node 2.
7. Using the SQL Server 2000 setup utility, upgrade the SQL Server 2000 default instance to a clustered instance on Node 2.

Upgrading Replication Servers and Subscribers

If you have deployed replication in your organization, you can still upgrade to SQL Server 2000. To successfully upgrade, you should follow a particular upgrade sequence to ensure that replication continues to operate successfully in your organization. The following sequence should be followed to successfully upgrade servers and subscribers that participate in replication:

1. Upgrade the distributor server first.
2. Upgrade the publisher server second.
3. Upgrade subscribers.

Immediate-updating replication enhancements in SQL Server 2000 will affect your replication configuration if you are using immediate-updating publications with snapshot or transactional replication. SQL Server 2000 uses a uniqueidentifier column instead of the timestamp column used in SQL Server 7.0 to track table updates. Additional changes to the triggers that help implement immediate-updating replication are also affected. The following upgrade procedures will allow you to support immediate-updating subscribers when upgrading to SQL Server 2000 replication:

1. Before replicating any data, upgrade both the publisher and subscribers.
2. Drop any publications that were enabled for updating and any current subscribers.
3. Update the tables on both the publisher and subscriber by dropping the timestamp column.
4. Recreate the publication and the subscribers to create the unique-identifier column and enable immediate-updating replication.

If your replication configuration employs File Transfer Protocol (FTP) for subscriber access, upgrading to SQL Server 2000 will turn off the FTP option. FTP parameters are stored in the Publication properties in SQL Server 2000, eliminating the need to define them for each subscription. After the upgrade process has completed, you will need to reset the FTP settings for each publication from the Publication Properties dialog box on the Snapshot Location tab.

Selecting a Recovery Model

The trunc. log on chkpt and select into/bulk copy options have been replaced with three recovery models in SQL Server 2000. To simplify recovery planning and backup and recovery procedures, you can select one of the recovery models shown in Table 1.3 from the database Properties dialog box on the Options tab.

Table 1.3: SQL Server 2000 Recovery Models

Model	Recovery Capabilities	Previous Version Settings
Simple	Recover up to the last successful backup. Remaining changes must be redone.	Trunc. log on chkpt: True Select into/bulkcopy: True or False
Full	Recover to any point in time.	Trunc. log on chkpt: False Select into/bulkcopy: False
Bulk-Logged	Recover up to the last successful backup. Remaining changes must be redone.	Trunc. log on chkpt: False Select into/bulkcopy: True

Each recovery model offers certain advantages for performance, space requirements, and data loss recovery. The *simple recovery* model requires the least amount of resources; log space is reclaimed because it is no longer needed for server recovery, similar to the trun. log on chkpt. option in previous versions of SQL Server. The disadvantage here is that data recovery to a particular point beyond the latest backup is not possible. Simple recovery should *not* be used where data recovery is critical and reentry is not possible.

Full and bulk-logged recovery models offer greater data recovery capabilities. *Full recovery* supports recovery to any point in time given that the current transaction log is not damaged. *Bulk-logged recovery* provides similar capabilities to the full recovery model, but bulk operations such as SELECT INTO, bulk loads, CREATE INDEX, image, and text operations are minimally logged, increasing your exposure to data loss in the event of a damaged data file. Selecting between full and bulk-logged recovery is dependent on your database structure and data operations. If complete data recovery is essential and the performance of bulk operations is not critical to your application, you should select the full recovery model for your database.

Summary

SQL Server 2000 represents the most robust database server that Microsoft has delivered to date. Building on the tremendous success of SQL Server 7.0, SQL Server 2000 enhances many of the previous version's popular features, including Data Transformation Services, OLAP (which has been renamed Analysis Services), and SQL Server's management and development tools. Numerous additions to SQL Server have been made to meet the scalability and availability demands of today's enterprisewide applications. Some of these significant improvements include:

- Federated servers, which provide support for database partitioning and updateable distributed partitioned views. Partitioning supports scale-out techniques, delivering practically boundless application scalability across commodity hardware.
- Large memory and SMP support, offering server utilization of up to 64GB of RAM and 32 processors for maximum scale-up potential.
- Enhanced OLAP and new data-mining capabilities for analyzing and forecasting activity and trends.
- Numerous programming and engine enhancements such as indexed views, user-defined functions, new data types, and trigger enhancements extend the capabilities of SQL Server. A redesigned Query Analyzer with SQL Debugger and an extended SQL Profiler offer strong and usable development tools.
- Multiple-instance support and database-level collation allow SQL Server to host and manage numerous autonomous applications on a single physical server.

As the first available member of Microsoft.NET Enterprise Servers, SQL Server 2000 provides native support for Extensible Markup Language (XML) and the Internet protocols HyperText Markup Language (HTTP) and Secure Sockets Layer (SSL). The .NET initiative grows on Microsoft's proven Distributed interNet Applications (DNA) Architecture Model by offering core support for industry standards to deliver application services inside and outside the enterprise.

Available in six different editions—Enterprise Edition, Standard Edition, Personal Edition, Developer Edition, Desktop Engine, and CE Edition—SQL Server 2000 provides a consistent programming model on every Windows platform from Windows CE through Windows 2000 Data-center Server. Designed for Windows 2000, SQL Server 2000 can take advantage of Active Directory integration for database and service publishing as well as security enhancements in Windows 2000. Windows CE Edition offers merge replication and remote data access for continuous application availability on connected and disconnected devices. This new level of platform supports opens the doors for SQL developers to deliver applications throughout the enterprise, on any device.

Backward compatibility with SQL Server 6.5 and 7.0 means nearly no immediate application changes, in most installations. Automatic upgrade capabilities allow SQL Server 7.0 installations to migrate to SQL Server 2000 with little impact on existing operations. Upgrade wizards migrate SQL Server 6.5 solutions to SQL Server 2000 in

little time. Multiple-instance support even allows you to make the transition from previous SQL Server versions to SQL Server 2000 on the same server.

To truly take advantage of SQL Server 2000's capabilities, migrating requires a defined and tested migration path. Identifying who and what will be affected by the migration is critical to its success. Analyzing your existing applications and determining where SQL Server 2000 can offer performance and availability enhancements make the difference between upgrading to SQL Server 2000 and taking advantage of SQL Server 2000. Training your administrators and support staff and executing test migrations in a lab scenario will not only help deliver a successful migration; these steps will allow you to design and modify your SQL applications for optimal performance and availability.

FAQs

Q. My Web application uses the n-tier architecture model of DNA. Will I have to redesign it to use SQL Server 2000 and .NET?

A. No and yes. SQL Server 2000 is, at its core, a relational database system, and like its previous versions, storing and managing data are what it does best. It has near 100 percent backward compatibility with SQL Server 7.0 and 6.5, so your applications will continue to work on SQL Server 2000, and they can immediately benefit from its performance and scalability improvements. To additionally benefit from SQL Server 2000, you should analyze your application and determine where it can take advantage of new capabilities such as indexed views, database partitioning, trigger enhancements, and the like. The bonus here is that SQL Server 2000 can coexist nicely with previous SQL Server versions, so you can migrate your databases and applications on your schedule.

As for .NET, to use the frameworks architecture that .NET offers, you need to rethink your application to take advantage of ASP+, ADO+, XML, and its Web services approach to application components. The plus side here is that .NET is an extension of the DNA model, not a replacement, so your DNA application can migrate to take advantage of .NET on your time schedule. .NET will not be finalized until early 2001, but there is an abundance of information on how these features will work, so visit Microsoft's .NET Web site at www.microsoft.com/net .

Q. Which version do I need if I want to distribute my Win32 application that uses a local, stand-alone SQL Server database?

A. The Microsoft SQL Server 2000 Desktop Engine (MSDE) is designed for the exact scenario you are asking about. MSDE is a freely distributable engine compatible with both SQL Server 2000- and Access 2000-based applications. Based on SQL Server 2000 Personal Edition, MSDE performance is limited to five users through a workload governor and does require licensing unless you connect your MSDE application to a SQL Server. Personal Edition would also work, but it is license bound to Enterprise or Standard Edition processor, and CAL licensing to distribution is restricted to your server licensing. MSDE includes no graphical tools, Analysis Services, or English Query.

Q. I had an Internet Connector license for my SQL Server 7.0 database and I want to upgrade to SQL Server 2000. Is there a comparable license plan? How do I upgrade?

A. Microsoft has implemented two licensing models, processor licensing and server/client access licensing, with the latest version of its server line. Processor licensing is required for an Internet/Web-connected database server to support unlimited connections. Starting with SQL Server 2000, there is no longer an Internet Connector License option for SQL Server.

Q. I am using SQL Server 2000 as the back-end database for my Web site application that is available on the Internet. What version and licensing do I need?

A. To support unlimited connections, SQL Server 2000 requires processor licensing. Upgrade pricing for licensing is approximately 50 percent off regular pricing. Upgrade pricing also includes competitive upgrades from products such as those

from Oracle and Sybase. The Internet Connector license option is no longer available; Internet Connector license owners may upgrade to processor licensing at regular upgrade prices.

Q. I would like to upgrade our SQL Servers to 2000, but we do not have additional resources for hardware and training. How should I approach this task?

A. This is a pretty common scenario in smaller organizations with restricted resources. SQL Server 2000 offers several upgrade options utilizing shared hardware. With multiple-instance support, you can run SQL Server 2000 side by side with your SQL Server 6.5 or 7.0 installations, so you can upgrade your databases on the same machine and then uninstall your previous version of SQL Server. Make sure to verify that your existing server hardware meets SQL Server 2000's recommended minimum requirements. As for training, there is nothing more important than training for a successful migration and a stable production environment. Although formal training programs can cost several thousand dollars, tools such as this book and your desktop computer will help you acquire the knowledge you need to support SQL Server. SQL Server 2000 can be installed on nearly every Windows platform, so you can begin to get familiar with its common features and additions without the expense of dedicated training equipment. Be sure to review the version comparison chart in this chapter so that you are aware of the restrictions of the version you can install on your training hardware.

Q. Our organization can benefit from Analysis Services, but we do not have a data warehouse or anyone who has designed or built a data warehouse. How do we get started?

A. Microsoft has done a tremendous job in making SQL Server 2000 Analysis Services usable by everyone. Numerous simple and advanced wizards will get you off and running with creating your first cubes, multidimensional expression statements, and data-mining projects. Although a data warehouse is a valuable asset, you can begin with your existing relational database. Review the Analysis Services chapter in this book; try the examples there and in the Analysis Services online help to get started.

Q. My applications can benefit from indexed views and user-defined functions. Is it possible to license certain features of SQL Server so I don't have to buy an Enterprise Edition license?

A. No. Microsoft has released three distinct versions, or editions, that are available for purchase: Enterprise Edition, Standard Edition, and Developer Edition. Each edition includes a subset of Enterprise Edition, which includes all SQL Server 2000 features, including indexed views. User-defined functions are a core SQL Server component and are included in all editions. Review the Edition Features Comparison table in this chapter for a detailed list of each edition's features.

Although it doesn't look as though it will happen any time soon, your idea of licensing specific features is one of the principles behind Microsoft's new commitment to "service"-based software. .NET's Web services are an introduction to this concept. Microsoft has made statements that it will make Microsoft Office available as a subscriber service in the future, and users will pay for the features that they need. Who knows—SQL Server 2010, pay for what you need?

Chapter 2: Installing and Configuring SQL Server 2000

Introduction

Whether your organization is migrating from an existing SQL Server installation, migrating from a different relational database system, or starting from scratch with its first database server, SQL Server 2000 is ready to help you get started. Strong support for previous versions of SQL Server make the upgrade process as straightforward as many traditional software upgrades. For migrations from SQL Server 6.5, the SQL Server Upgrade Wizard will provide a clean switch to SQL Server 2000. The Database Copy Wizard and Data Transformation Services can assist you in transferring database solutions from other database servers to your new SQL Server at any point after your initial SQL Server installation has been completed.

SQL Server 2000 offers new features that will benefit the increasingly popular role of application service providers (ASPs) and large-scale deployment tasks. Aside from the ability to generate answer files for performing multiple automated installations, SQL Server 2000 now supports disk imaging. SQL Server installations can be disk imaged and distributed to multiple servers. This feature allows organizations to deploy identical SQL Server configurations on demand.

Another popular installation option in SQL Server 2000 is its support for multiple instances. With each instance operating comparable to an independent server, multiple instance support will allow organizations such as application service providers and Web-hosting providers to offer dedicated SQL Servers without managing independent physical servers. Tasks such as client access, administration, and security concerns are minimized when compared with single database servers hosting multiple applications.

The following sections walk you through planning your upgrade or clean installation of SQL Server 2000. You will discover SQL Server 2000's hardware and software requirements as well as its new licensing model. At the end of this chapter, you will be able to select an appropriate storage system and disk configuration, install and upgrade to SQL Server 2000, install additional SQL Server components, and configure multiple instance support as well as review SQL Server's configuration options.

Backing Up a System Prior to Installing SQL Server 2000

A quick and easy way to back up a system prior to installing SQL Server 2000 is to use a disk imaging tool such as Symantec's GHOST, DRCopy, or PowerQuest's Drive Image Product. Backing up the system allows for a faster restore if the installation fails; it returns the drive to its state before the installation.

These tools take a “snapshot” of the disk’s contents, including registry, data, and operating system files. This snapshot allows for fast recovery from a failed installation.

One caveat when using imaging tools: You need to be sure you have enough space to store the partitions or disks you are backing up. Most of the drive imaging tools allow for the image to be written in chunks spanning multiple disks or files. This feature is especially useful if you want to burn the image onto CDs or DVDs. It is also very important to consider network capacity if you are backing up to a network drive.



Planning a SQL Server Installation

Because the SQL Server 2000 installation has become much easier, you will be tempted to install or upgrade without considering the planning necessary to ensure a smooth installation or upgrade. When planning for a SQL Server installation, don't overlook your backup strategy. The best fault-tolerant systems can fail, and accurate backups will be priceless at that point in time.

Some things you should consider prior to installing SQL Server 2000:

- Be sure you have appropriate rights on the machine on which you are installing Microsoft SQL Server 2000. You don't need to be a domain administrator, but you do need administrative rights on that machine.
- Run the appropriate Database Console Commands (DBCC) on the existing SQL Server databases to ensure that they are in a consistent state.
- Be sure that the computer meets the system requirements for Microsoft SQL Server 2000. For more information, refer back to Chapter 1 .
- Back up SQL Server if you are installing SQL Server 2000 on the same computer. Wherever possible, install and test on a nonproduction system first. Use a test environment that matches your production environment. Doing so can save you a lot of stress and time!
- If you are installing a cluster, you must disable NetBios on all network cards prior to installing SQL Server 2000.
- Check the SQL Server installation options and be sure that you know the options you plan to select. Getting halfway through the installation and not knowing what choice to make could result in a failed installation!
- Be sure you have thought out the file locations to which you are going to install SQL Server 2000 and your data files.
- Consider storage requirements prior to beginning your installation. In most cases, you need a fault-tolerant storage system.
- If you are planning to use regional settings other than English, make sure you understand the implications of this change.
- If you are planning to change the character set or collation settings, make sure you understand the implications of this change.
- Consider performance prior to beginning your installation. Performance is dependent on processor speed, the number of processors in the system, disk I/O throughput, and memory.

SQL Server 2000 can co-exist with SQL Server 7.0; however, two copies of SQL Server loaded in memory at the same time result in degraded performance.

Note When selecting a machine on which to install SQL Server 2000, if the machine is a domain controller, the person installing SQL Server 2000 needs to be a domain administrator because domain controllers do not have local user accounts.

For this reason, care must be exercised in selecting a domain controller to run SQL

Server. Installing SQL Server on a domain controller is generally considered bad operating practice and is not recommended. Domain controllers should be left to perform their network-related tasks.

Installation Requirements

This section outlines the minimal requirements for SQL Server 2000. Keep in mind that these are the minimum requirements; in most cases, these will not suffice for a production environment.

Hardware Requirements

SQL Server 2000 is very flexible with regard to hardware requirements. SQL Server 2000 will run on most current Microsoft operating systems. SQL Server 2000 requires the following hardware.

Computer/Processor Requirements:

- The computer must be an Intel or compatible: Pentium 166MHz or higher, Pentium Pro, Pentium III, or the minimum processor required for your operating system.
- Some versions of the Cyrix processor are not compatible with SQL Server 2000 because they don't contain the full Intel set of instructions.

Memory Requirements:

- Enterprise Edition: 64MB minimum, 256MB recommended
- Standard Edition: 32MB minimum, 128MB recommended

Hard Disk Requirements:

- SQL Server 2000, full install: 180MB
- SQL Server 2000, typical install: 170MB
- SQL Server 2000, minimum install: 65MB (minimum)
- SQL Server 2000, client tools only: 90MB
- Analysis Services: 50MB
- English Query: 12MB

Software Requirements

SQL Server 2000 software requirements depend on the edition of SQL Server you are installing. Table 2.1 outlines the software requirements for SQL Server 2000.

Table 2.1: SQL Server 2000 Software Requirements

Edition	Operating Systems Supported

Table 2.1: SQL Server 2000 Software Requirements

Edition	**Operating Systems Supported**
Enterprise Edition	Microsoft Windows NT Server 4.0 with Service Pack 5 or later
	Windows NT Server 4.0 Enterprise Edition with Service Pack 5
	Windows 2000 Server
	Windows 2000 Advanced Server
	Windows 2000 Datacenter Server
Standard Edition	Microsoft Windows NT Server 4.0 with Service Pack 5 or later
	Windows NT Server 4.0 Enterprise Edition with Service Pack 5 or later
	Windows 2000 Server
	Windows 2000 Advanced Server
	Windows 2000 Datacenter Server
Personal Edition	Microsoft Windows 98
	Windows Me
	Windows NT Workstation 4.0 with Service Pack 5 or later
	Windows NT Server 4.0 with Service Pack 5 or later
	Windows NT Server 4.0 Enterprise Edition with Service Pack 5 or later
	Windows 2000 Professional Personal Edition
	Windows 2000 Server
	Windows 2000 Advanced Server
	Windows 2000 Datacenter Server
Developer Edition	Microsoft Windows NT Workstation 4.0 with Service Pack 5 or later
	Windows NT Server 4.0 with Service Pack 5 or later
	Windows NT Server 4.0 Enterprise Edition with Service Pack 5 or later
	Windows 2000 Professional
	Windows 2000 Server
	Windows 2000 Advanced Server
	Windows 2000 Datacenter Server
	Windows 2000 Terminal Server
Desktop Engine	Microsoft Windows 98
	Windows Me
	Windows NT Workstation 4.0 with Service Pack 5 or later

Table 2.1: SQL Server 2000 Software Requirements	
Edition	**Operating Systems Supported**
	Windows NT Server 4.0 with Service Pack 5 or later
	Windows NT Server 4.0 Enterprise Edition with Service Pack 5 or later
	Windows 2000 Professional
	Windows 2000 Server
	Windows 2000 Advanced Server
	Windows 2000 Datacenter Server
Windows CE Edition	Windows CE devices

SQL Server Licensing

One of the changes between SQL Server 7.0 and SQL Server 2000 is the licensing model. With SQL Server 2000, Microsoft has introduced hardware-based licensing. SQL Server 2000 offers per-processor licensing and per-seat licensing. Each of the types of licenses is reviewed in this section.

Processor License

Under the per-processor license model, each processor running SQL Server requires a license. That is to say, a computer containing four processors requires four processor licenses. Any number of computers can connect using processor-based licensing. This includes Internet connections, extranets, intranets, and so on. No additional licenses are required in using this model.

Server License and Client Access License

Under the per-seat licensing model, a client access license is required for each client that will connect to the database. In addition, each server needs to have a license to run the SQL Server software. A per-seat license allows the client to connect to any SQL Server on your network.

More and more applications are being built using the *n-tier* architecture. These installations present a scenario in which the clients connect to the Transaction Layer as opposed to the SQL Server. In instances in which Microsoft Transaction Server or some other middleware product is used between a client and the SQL Server, there is still an indirect connection between the client and the SQL Server, so a client access license is still needed.

Installation Options

This section covers the installation options that are available to you for installing SQL Server 2000. There are many options for installing SQL Server 2000. The decisions you make during the installation will affect the way SQL Server behaves, so understanding the installation options and configuration choices is important before you begin the installation process.

Local vs. Remote Installation

SQL Server 2000 offers both local and remote installation options. Local options install SQL Server 2000 on the local machine; remote options install SQL Server 2000 on a remote machine.

In order to properly install SQL Server on a remote machine, it's necessary that you have administrative rights on the remote machine. Be sure you have the appropriate rights to the computer on which you are installing SQL before you begin the installation process.

Disk Systems

Choosing the right disk system for your database is very important. A 200GB drive stores as much data as 10 20GB drives, but the performance can be one-tenth of the 10-drive array due to resource contention or device strain. Take, for example, a single disk server configuration running multiple disk-intensive applications such as SQL Server. If each of these applications is continually requesting read and write activity from the disk, requests will be queued up while the disk system fulfills them. This contention of resources will degrade the performance of all the applications that are requesting disk activity. Device strain, on the other hand, is simply overloading a hardware device such as a disk controller. A large OLTP system can continuously make thousands of read and write requests per minute. If the database server were utilizing low-cost integrated development environment (IDE) drives or even single Small Computer Systems Interface (SCSI) disk configurations, the application would quickly degrade to unacceptable performance levels.

With the eventual performance and success of your database applications in mind, choosing a reliable and high-performance disk system is essential to your database server. You should always use SCSI disks because of their performance and throughput. Production servers should also use multiple SCSI disks in a RAID configuration for both performance and reliability.

Note Hardware RAID generally outperforms software RAID. In software-based RAID, CPU cycles are used in order to perform RAID functions, and this affects application performance. Hardware RAID typically offers other features such as the ability to *hot swap* disk drives, or change drives without powering off your server. Software RAID is cheaper and might be sufficient on installations with sufficient CPU capacity.

RAID

RAID , or *redundant array of inexpensive disks,* can provide both a performance increase and fault tolerance. RAID level 5 is the most commonly used form of providing fault tolerance for database systems. RAID can be implemented in hardware or software. Hardware RAID provides the fastest performance and reliability.

RAID drives appear to the application as one logical drive. With a RAID drive, data is distributed across multiple drives, so the data can be written simultaneously. This feature reduces the bottleneck of single disk configurations.

RAID 0, or disk striping, offers a performance increase but no fault tolerance. RAID 0 distributes data across all the disks in *stripes* . This method takes advantage of distributing disk I/O across disks for a logical read or write. For example, if a disk's read rate is 100 bytes/sec (bps) by striping the data across five disks, you could get an effective read rate of 500bps. The major drawback to RAID 0 is that the failure of just one drive will result in the loss of all data in an array. RAID 0 should never be used in production environments unless loss of data can be tolerated.

RAID 1 is disk mirroring. In RAID 1, a mirror copy of a disk is created and is simultaneously written to and read from a compatible RAID controller. This method offers no performance gain but does offer fault tolerance by duplicating data across disks. In the event that either disk fails, the second disk in the mirror set can be used to continue servicing disk requests and recover the damaged disk.

RAID 2 through *RAID 4* exist but are seldom used in production environments today. These provide varying degrees of performance and data integrity.

RAID 5 is the most commonly implemented solution in production environments. It provides a high level of performance along with a high level of data integrity. RAID 5 requires a minimum of three physical disks for the array. RAID 5 can sustain one disk failure and continue to function. It writes parity information across the disks, allowing the data to be recreated and read, even if one disk fails. After the damaged disk is replaced, the data on it can be recreated to continue functioning in the RAID set.

RAID 6+ exist but are not yet widely used in production environments today. These versions provide varying degrees of performance and data integrity.

Storage Area Networks

The concept of a *storage area network (SAN)* involves providing access to shared storage subsystems on a single network or spanning multiple networks to back up, store, and share data.

The servers are typically connected to very large storage devices by a fast connection. The servers communicate by emulating the (SCSI) protocol over TCP/IP and require special management software. With this technology, servers and storage devices can span wide area network (WAN) links to various geographic locations.

Creating Service Accounts for SQL Server

SQL Server service accounts allow SQL Server to run with the rights and privileges assigned to the service account. This is better than using an existing user's account, because if the password on the account is changed, it is necessary to change the password in SQL Server 2000. This is easily done through properties in Enterprise Manager but will cause SQL Server to fail to start.

When running under Windows NT or Windows 2000, SQL Server and SQL Server Agent run as services. These can be viewed, started, and configured under the Services applet in Control Panel.

SQL Server 2000 and SQL Server Agent require a user account to run. These can be run under any user account that has the appropriate access, but general practice is to assign them their own service accounts. Typically, SQL Server and SQL Server Agent are assigned the same user account, either the local system or domain user account, but this is not required. In order for interserver processes to work smoothly, it's best to use a domain account.

It's good practice to create the service accounts prior to beginning to install SQL Server.

The local system account is a built-in account that doesn't require a password. This account has no network access that will limit the ability of SQL Server to communicate with other SQL servers on the network. Generally, it is preferred to use a domain service account in instances in which the SQL Server is on a network.

Note Since Windows 98 doesn't support services, SQL Server and SQL Server agent "simulate" a service account, it is not necessary (nor is it possible) to create a service account in Windows 98.

Using a Domain User Account

The domain user accounts used by SQL Server 2000 use Windows authentication, like any other user account. This is necessary for interserver communication such as:

- Replication
- Remote backup strategies
- Cross-server joins
- SQL Agent jobs
- SQL Mail

More than one server can use the domain user account. When configuring security for servers that are using replication, the it is recommended that a Publisher and all its Subscribers share the same service account for the SQL Server service.

Requirements for Domain User Account

All domain user accounts for SQL Server 2000 must have the following permissions:

- Change level access to the SQL Server directory (\Program Files\ Microsoft SQL Server\Mssql).
- Change level access to the .mdf, .ndf, and .ldf database files.
- The ability to log on as a service.
- The ability to read and write registry keys at and under the following registry hives:
    - HKEY_LOCAL_MACHINE\Software\Microsoft\MSSQLServeror for any named instance: HKEY_LOCAL_MACHINE\Software\Microsoft\ Microsoft SQL Server
    - HKEY_LOCAL_MACHINE\System\CurrentControlset\Services\ MSSQLServer or for any named instance: HKEY_LOCAL_MACHINE\ System\CurrentControlset\Services\MSSQL$Instancename
    - HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\ CurrentVersion\Perflib

In addition, a domain user account must be able to read and write corresponding registry keys for SQLAgent$InstanceName, MSSearch, and MSDTC services.

Changing User Accounts

To change the password or other properties of any SQL Server-related service after installing SQL Server, use SQL Server Enterprise Manager. If your Windows password expires and you change it, be sure to also revise the SQL Server Services settings in Windows.

Disk Imaging Support

SQL Server 2000 includes support for creating a *disk image* for distribution. A disk image is a convenient way to deploy SQL Server 2000 to other machines. With this method, you install SQL Server on a machine, stop the service, and at that point use that as your image. When the server restarts, if the machine name has changed (because the image is now on a different machine), SQL Server realizes that it is a different machine and it will correct the name in SQL Server.

It's important to note that this can be done only the first time SQL Server is restarted. After that, the name in SQL Server will not change.

Answer File for Automated Installations

SQL Server 2000 provides a way to automate installation by using unattend files. These files answer the questions you are asked on the screen. This is useful if you need to create several SQL Server 2000 installations configured the same way. There are a number of ways to create the Setup file. The easiest is to use the Unattend file (setup.iss).Additionally, when setting up SQL Server, you can simulate an installation

and create the answer file by selecting Record Unattended .ISS File from the advanced installation (see Figure 2.1).



Figure 2.1: The Advanced Options screen.

Note When you install SQL Server using the Setup Wizard, your actions are recorded into Setup.iss in the system root directory (\winnt in a default Windows NT or Windows 2000 installation). When manually changing an automated answer file, always start with the answer file that most closely matches the goal you are currently trying to accomplish.

Multiple Server Instances

SQL Server 2000 supports multiple "instances" of SQL Server 2000 on a single server. An instance is similar to a virtual SQL Server—it has its own name, its own set of passwords, and its own settings that can be different from the other instances on a particular machine. Instances can have different collation and language settings as well. This allows for quite a bit of flexibility with SQL Server.

Only one instance on a machine will be the *default instance* . The default instance behaves as a SQL Server 6.5 or 7.0 installation with respect to connectivity. Applications that are set up to use SQL Server 6.5 or 7.0 databases will usually connect to the default instance with little or no modification.

The other instances on the computer are referred to as *named instances* . Each instance must have a unique name associated with it.

Instances also provide the ability to isolate databases even further. An instance can be stopped and started without affecting other SQL Server instances on the same machine. This is very helpful when bringing a new application online.

Another benefit to having multiple instances on a machine is the ability to have a *test bed* of instances for development purposes. An application developer can very quickly compare how database changes will affect an application, without having to have a separate machine for development and quality assurance.

Having multiple instances of SQL Server 2000 can also help with the development and debugging of international applications. An instance with the character set for each country can exist on a machine in development, and the developer can test on each instance without rebuilding or requiring additional hardware.

Finally, having multiple instances can reduce hardware requirements. Having one powerful machine with 10 instances is much easier to support than 10 less powerful servers.

Collation Options

Collation refers to the rules as to how SQL Server sorts and compares data. In SQL Server 2000, you can specify a named collation. Case sensitivity, accent marks, kana character types, and character width are all determined by the collation options selected when installing SQL Server.

Two kinds of collations are used by SQL Server: Windows collations and SQL collations. Windows collations define the collation set based on the Windows locale. SQL collations are provided more for backward compatibility with older SQL databases.

Tip Be sure to decide on your collation options prior to installing SQL Server 2000. Even though you can change your options after installation, doing so requires a complete rebuild of your databases. Additionally, it is recommended that you develop a standard within your organization for these options. Cross-server activities can fail if the collation settings are not consistent across servers.

Upgrading to SQL Server 2000

Upgrading from SQL Server version 7.0 is done via a menu option on the installation screen for SQL Server 2000. When you choose to upgrade an existing version of SQL Server, the installation program determines the existing installed version and walks you through the correct machines. Keep in mind prior to installing SQL Server 2000 that you cannot easily go back to Version 6.5 or 7.0; to do so requires removing SQL Server 2000 and reinstalling SQL Server 6.5 or 7.0, then restoring from backup.

Some of the options offered for installing SQL Server are:

- A complete installation upgrade from SQL Server 7.0 to SQL Server 2000; this process will overwrite SQL Server 7.0
- Adding components to an installation of SQL Server 2000
- Adding features to the existing version of SQL Server 2000
- Upgrading the edition of SQL Server 2000 (for example, from Standard to Enterprise)
- Upgrading SQL Server version 6.5 to SQL Server 2000 using the wizard
- Upgrading an existing SQL Server 7.0 online database using the Copy Database Wizard

As part of the upgrade from SQL 7, the external tools such as the Management Console and Query Analyzer will be replaced by the SQL Server 2000 versions. The Microsoft Distributed Transaction Coordinator (DTC) will also be upgraded. Note that as in most Microsoft installations, a fail-safe mechanism keeps the previous version around until the installation is complete.

SQL Server 2000 Coexisting with Previous Versions of SQL Server

SQL Server 2000 can coexist with previous versions of SQL Server. The new instances of SQL Server 2000 are installed as named instances, whereas the previous SQL installation will be the default instance.

Upgrading from SQL Server 7.0 to SQL Server 2000

Upgrading from SQL Server 7.0 is the easiest of upgrades. SQL Server 7.0 will be automatically overwritten with SQL Server 2000 if an upgrade is selected. The data in the SQL Server 2000 database will be upgraded to SQL Server 2000 format. The tools and program will be replaced with the SQL Server 2000 versions, but the SQL Server 7.0 Books Online will be left.

Any SQL Server 7.0 profiler traces you have created will not be upgraded when SQL Server 7.0 tools are upgraded to SQL Server 2000. In addition, any information models that were installed with Microsoft Repository 2.0 will not be upgraded automatically. As part of the upgrade, you can also upgrade from one edition to another edition of SQL Server 2000. After you upgrade, you should run update statistics and repopulate full-text catalogs to be sure SQL Server 2000 will perform optimally.

Tip Be sure you are ready to upgrade to SQL Server 2000 prior to installing it; the only way to go back is to remove SQL Server 2000 and reinstall SQL Server 7.0 (and any SQL Server service packs), then restore the database from the backup.

Upgrading from SQL Server 6.5 to SQL Server 2000

When you upgrade from SQL server 6.5 to SQL Server 2000, the installation program will guide you through the process. Prior to beginning the upgrade, make sure of the following:

- You have enough free disk space; you need 1.5 times the space for data.
- You have backed up your existing SQL Server.
- You are ready to upgrade; the only way back is to remove SQL Server 2000 and reinstall SQL Server 6.5.
- SQL Server 2000 is the default instance on the computer on which you are installing.

When upgrading from SQL Server 6.5, be sure to repopulate the full-text catalog and update statistics after the upgrade to ensure optimal performance of SQL Server 2000.

Tip Be sure you have enough disk space when you upgrade from SQL Server 6.5. In addition to the hard disk space used by Microsoft SQL Server 2000, you need approximately 1.5 times the size of the SQL Server 6.5 databases.

Upgrading Replication Servers

If you are upgrading servers used in replication, you need to upgrade the servers in a specific order. The order for replication servers is as follows:

1. Distributors
2. Publishers
3. Subscribers

Following this order is necessary to be able to continue to use replication when the servers are running different versions of SQL Server 2000. If you are using immediate updating with snapshot replication or transactional replication, there are other issues you need to consider: You need to drop all publications and subscriptions and republish and resubscribe after all machines are upgraded. There are also issues with upgrading on clusters with replication. You need to uncluster the SQL Servers prior to upgrading them.

Installing SQL Server

In this section, we discuss installing SQL Server 2000. Please be sure you have read and understood the planning section prior to installing SQL Server.

Standard Installation

Let's now step through a SQL Server 2000 installation. A simple installation can be performed in around 30 minutes. To begin installing SQL Server 2000, put the CD in the CD-ROM drive and run Autorun, then select SQL Server components unless you are installing on Windows 95. In that case, you might need to install prerequisites. The main installation screen is shown in Figure 2.2 .



Figure 2.2: The Main Installation screen for SQL Server Standard Edition.

Early in the installation process, you will be asked if you are performing a local or a remote installation. If you select remote installation, you will be prompted to select the server on which you want to install (see Figure 2.3).

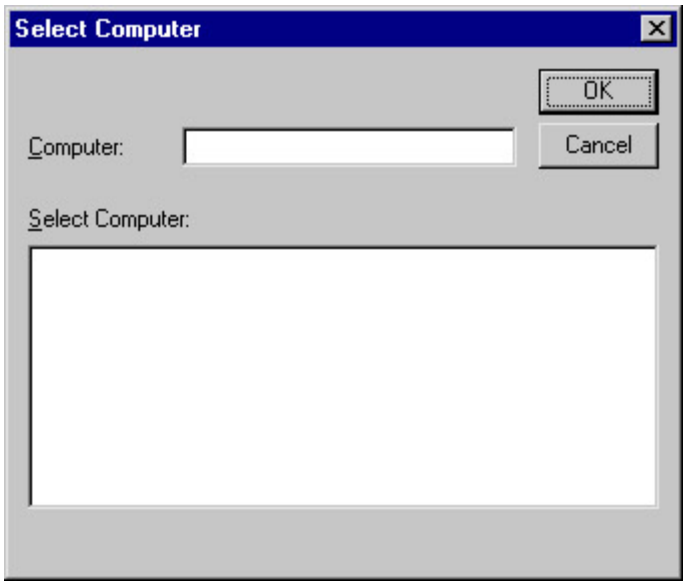


Figure 2.3: The Select Computer dialog box.

When you begin your installation, you will be prompted to create a new instance of SQL Server or to upgrade an existing SQL Server database (see Figure 2.4). Select upgrade, remove, or add if you want to change features; upgrade an existing SQL Server installation; or remove an installation. Advanced is used for fail-over clustering, rebuilding the registry or to create an ISS file without installing SQL Server 2000.



Figure 2.4: The Installation Selection screen.

For a new instance or to install client tools or connectivity, make the appropriate choice on the Installation Definition screen (see Figure 2.5).



Figure 2.5: The Installation Definition screen.

Additionally, you will be asked to name the instance you are creating. The named instance you create can be addressed from your application. In Figure 2.6 , an instance named SQL2000_test will be created. If you were installing the default instance, the default box would be checked.

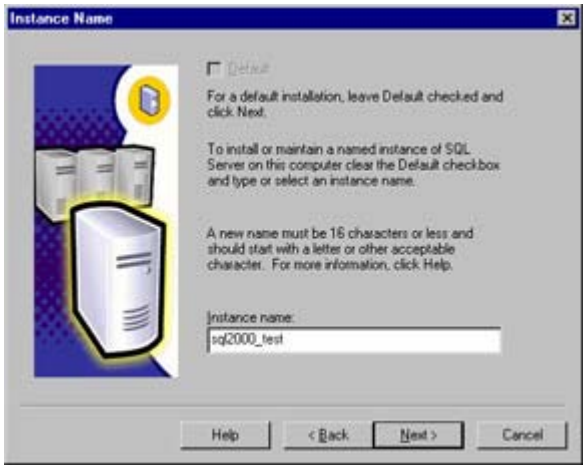


Figure 2.6: The Instance Name screen.

Figure 2.7 shows where the database files and application files will be installed. The location of the data files is critical; be sure you select where you want to install the data files at this point.

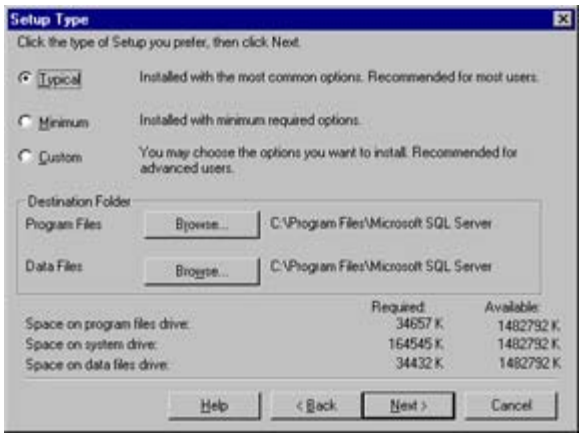


Figure 2.7: The Setup Type screen.

If you choose custom installation, the Select Components screen will be presented to you. It allows you to choose the SQL Server 2000 components to install (see Figure 2.8).

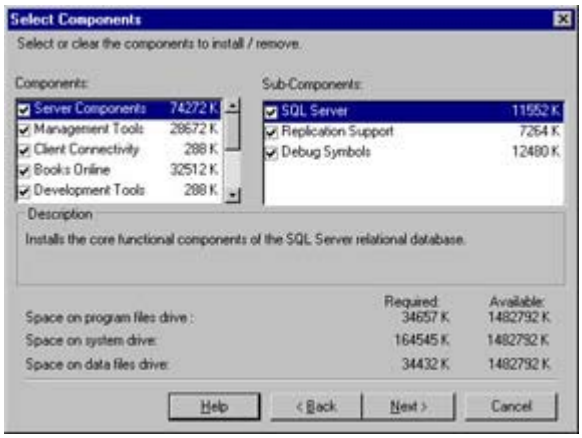


Figure 2.8: The Select Components screen.

In the Services Accounts screen, you will be prompted to select an account under which SQL Server will run. In most cases, this should be a domain account (see Figure 2.9).

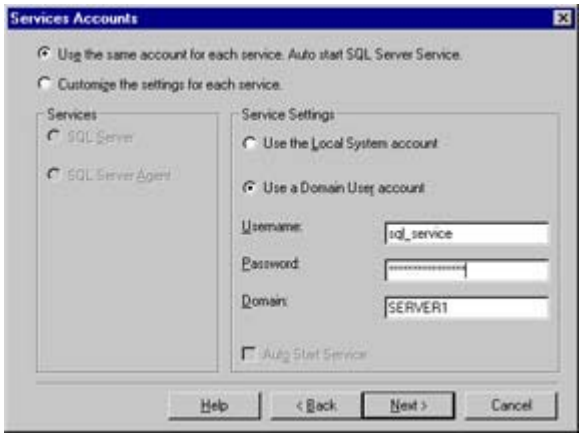


Figure 2.9: The Services Accounts screen.

The Authentication Mode screen (see Figure 2.10) allows you to choose a password for the SQL system administrator (sa) account. It is very important to protect this account; the SA account has complete control over the SQL Server instance. Be sure to choose a password that cannot be easily decoded.



Figure 2.10: The Authentication Mode screen.

SQL Server will prompt you for the licensing mode you are using (see Figure 2.11). It is important to select the proper licensing mode. For more information, see the "licensing mode" in the planning section.

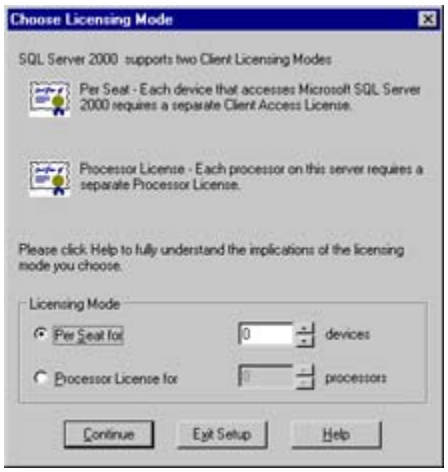


Figure 2.11: The Choose Licensing Mode screen.

Once setup is complete, you might need to reboot the computer on which you are installing SQL Server. This is one of the reasons it's not a good idea to install SQL Server on a domain controller.

Advanced Installation

When you select the Advanced option in the Installation Options Setup screen, the Advanced Options dialog box provides three choices:

- **Record Unattended .ISS File** This option allows you to create a setup initialization file for unattended installations.
- **Registry Rebuild** This option rebuilds the registry for a corrupted installation.
- **Maintain a Virtual Server for Fail-over Clustering** Make changes to existing clusters, such as revising the name or adding and removing cluster nodes.

Configuring Cluster Support

A cluster consists of a *virtual server* , which consists of one or more disks and one or more servers. A fail-over cluster generally runs on two or more servers.

On the Setup screen, you will select Virtual Server and type in the name you are using for your virtual server. The installation will step you through the clustering screens, allowing you to choose a disk cluster as well as other pertinent information about the cluster installation. Maintaining a virtual server for fail-over clustering is done from the Advanced Options installation tab (see Figure 2.12).



Figure 2.12: The Advanced Options screen.

Unattended Installation

Unattended installation provides a convenient way to install SQL Server 2000 on machines with a similar configuration. Unattended installation can also aid in installing SQL Server 2000 on machines in remote offices. Someone who is not familiar with SQL Server 2000—maybe an office administrator—can perform unattended installation. Note that you can't perform an unattended installation to set up a fail-over cluster of Microsoft SQL Server 2000.

It is possible to run Setup and not install SQL Server 2000 but create an ISS file based on your responses to the wizards. This process will simulate an installation, but no files will actually be copied. You can access this feature from the Advanced Options tab during setup. It is then quite simple to modify the .ISS file with your favorite text editor to customize it for other machines.

The SQL Server 2000 CD contains some sample .ISS files; along with batch files, you can use these as templates to create your own .ISS file and batch file. These files can be found in the root directory of the CD.

Configuration Options and Settings

In this section, we discuss the various configuration options and settings. Configuration settings can have a dramatic effect on SQL Server performance as well as other applications running on the same server.

SQL Server Properties

In this section, let's step through each of the settings for SQL Server 2000. We discuss the setting and why you would (or would not) want to adjust it.

We start by examining the General tab (see Figure 2.13). The General tab provides information about the installation and the computer, such as installed memory, collation, and the like. Additionally, you can select the parts of SQL Server to autostart with the operating system. You can choose SQL Server, SQL Server Agent, and MS DTC. Generally, you want SQL Server to autostart when the operating system starts. One of the few exceptions to this rule is if you have a machine that is used for only occasional SQL Server development; in this case, you might want to start SQL Server manually. If you are not using DTC and SQL Server Agent, you would not want to start them, because they consume resources on the machine when they are running. Note that MS DTC and SQL Server Agent will be used in most production environments.

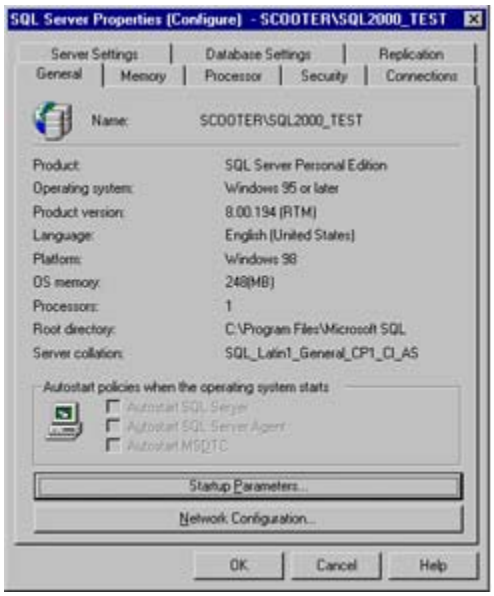


Figure 2.13: The SQL Server Properties General tab.

Clicking the Startup Parameters button will cause the Startup Parameters dialog box to appear (see Figure 2.14). This box allows you to customize the startup parameters for SQL Server 2000.

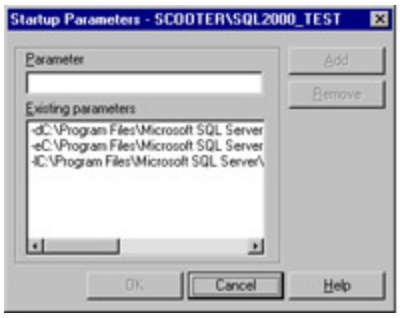


Figure 2.14: The Startup Parameters dialog box.

Table 2.2 summarizes the startup options available for SQL Server 2000.

Table 2.2: Startup Options Available for SQL Server 2000

Option	**Description**
- **d** *master_file_path*	The full path and filename to the master database file.
-e *error_log_path*	The full path and filename to the error log file.
-l *master_log_path*	The full path and filename to the master log file.
-c	Allows SQL Server to start faster by not running as a service.
-f	Starts SQL Server using minimal resources.
-g	Allows you to adjust SQL Server memory allocation internally to specify memory within the SQL Server process but outside the memory pool.
-m	Starts SQL Server in single-user mode.
-n	Do not use Windows Event Log for SQL Server events and errors.
-s	Start a named instance of SQL Server.
/T *trace#*	Start SQL Server using a trace flag for “nonstandard” SQL Server behavior.
-x	Disables the keeping of cache hit statistics; can sometimes improve SQL Server performance.

Clicking the Network Configuration button allows you to configure the network protocols for SQL Server to use. In Figure 2.15 , we are using TCP/IP. NWLink IPX/SPX is disabled. It’s a good idea to disable the network protocols you are not using. Enabling unused protocols can result in a performance decrease. Protocol encryption increases security at a performance cost. Use this option only if it is necessary.

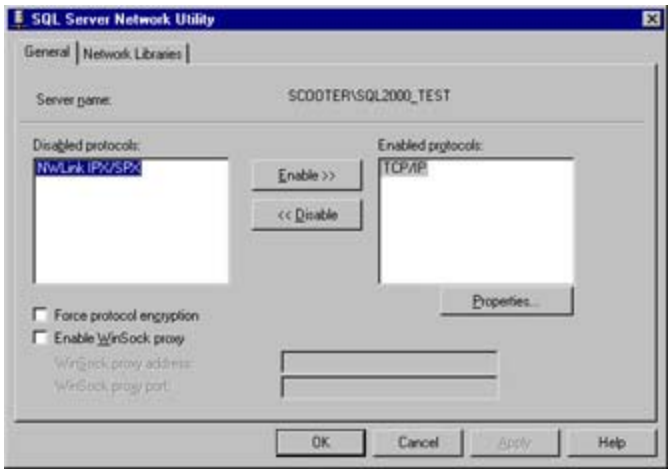


Figure 2.15: The SQL Server Network Utility screen.

The Memory Tab

Memory can be adjusted from the Memory tab. There are a few reasons you might want to adjust memory usage. One reason is to restrict the memory SQL Server uses on installations in which other memory-intensive applications are installed on the server. Additionally, you might want to reserve a minimum amount of memory for SQL Server if other memory-intensive applications are installed (so the other memory will always be available for SQL Server).

Using the Dynamically Configure SQL Server Memory Settings sliders, you can set minimum and maximum memory usage. In Figure 2.16 , SQL Server 2000 is configured to dynamically use 0 to 248 megabytes of memory. If we were to use a fixed memory size, 124 megabytes of memory would be used. The slider with the colored bar above it allows us to adjust this value up and down.

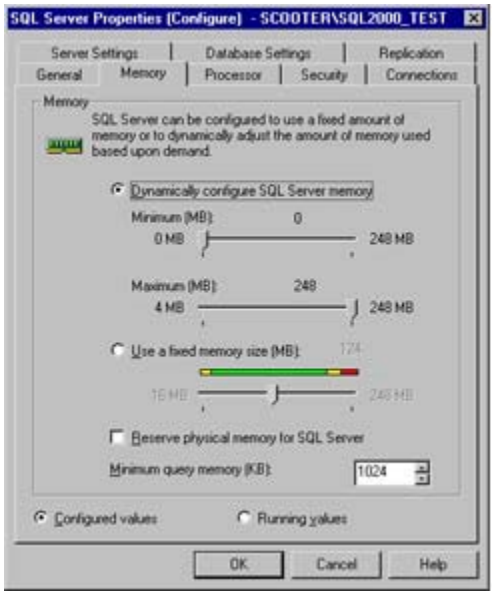


Figure 2.16: The SQL Server Properties Memory tab.

The “Minimum query memory” option specifies the minimum amount of memory that can be allocated per user for query execution. The default is 1024 kilobytes (KB). The Configured Values option allows you to configure the options for the server. The Running Value option button allows you to view the current running values for the options on this tab. These values are read-only.

The Processor Tab

On the Processor tab, you can control the processors SQL Server will use. This can be useful if other applications are running on a machine and you want to throttle processing back on SQL Server (see Figure 2.17).

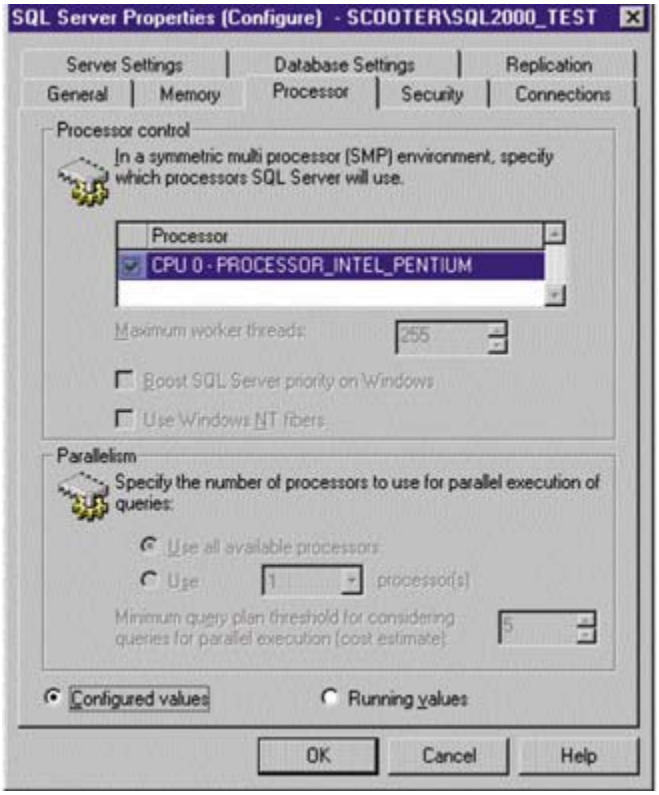


Figure 2.17: The SQL Server Properties Processor tab.

Boosting SQL Server priority on Windows is useful to improving SQL Server performance but will affect the performance of other applications running on the same server. This feature should be used only on computers dedicated to running only SQL Server. Process priorities in Windows NT and Windows 2000 are assigned a number, with base priority 7 on a single processor machine and 15 on a multiprocessor machine. The higher the priority, the more processor time allocated to the application. In Windows, the priority levels range from 0 (lowest priority) to 31 (highest priority).

Priorities are applied to threads by combining the priority class of the process with the priority level of the thread.

Selecting “Use NT fibers” tells SQL Server that you want an instance of SQL Server to use fibers instead of threads. When using fibers, SQL Server allocates one thread per CPU and then allocates one fiber per concurrent user, up to the maximum worker threads value. This setting takes effect after you restart the server.

The principle difference between threads and fibers is that fibers are managed by the application, whereas threads are managed by the kernel. Using fibers generally doesn’t offer an advantage unless you are running on a high-volume multiprocessor machine, in which fibers could offer a performance increase.

“Use all available processors” specifies that you want SQL Server to use all available processors for the parallel execution of queries. This option can be useful in throttling back the processing SQL Server 2000 will use.

“Use processors” allows you to specify the number of processors you want SQL Server to use for the parallel execution of queries. This option also can be useful in throttling back the processing SQL Server 2000 will use.

“Minimum query plan threshold” for considering queries for parallel execution allows you to specify the threshold at which SQL Server creates and executes parallel plans. SQL Server creates and executes a parallel plan for a query only when the estimated cost to execute a serial plan for the same query is higher than the value set for this option.

The Configured Values option button allows you to configure the options for the server. The Running Value option button allows you to view the current running values for the options on this tab. These values are read-only.

The Security Tab

On the Security tab, in the Authentication section, the SQL Server and Windows option button specifies that users can connect to the instance of SQL Server 2000 using SQL Server Authentication and Windows Authentication. This is considered Mixed Mode authentication. Users who connect through a Microsoft Windows NT 4.0 or Windows 2000 user account can use trusted connections in either Windows Authentication or Mixed Mode.

The Windows Only option button specifies that users can connect to the instance of SQL Server using Windows Authentication only.

Under the Auditing section, selecting None disables auditing. This is the default for this setting.

Under the Auditing section, selecting Success audits successful login attempts. You can record attempted user accesses as well as other SQL Server log information and enable auditing for both security modes. You can also record information on both trusted and distrusted connections. Log records for these events appear in the Microsoft Windows

application log, the SQL Server error log, or both, depending on how you configure logging for the instance of SQL Server. If you select this option, you must stop and restart the server for auditing to be enabled.

Under the Auditing section, selecting Failure audits failed login attempts (see Figure 2.18). You can record attempted user accesses as well as other SQL Server log information and enable auditing for both security modes. You can also record information on both trusted and distrusted connections. Log records for these events appear in the Windows application log, the SQL Server error log, or both, depending on how you configure logging for your instance of SQL Server. If you select this option, you must stop and restart the server to enable auditing.

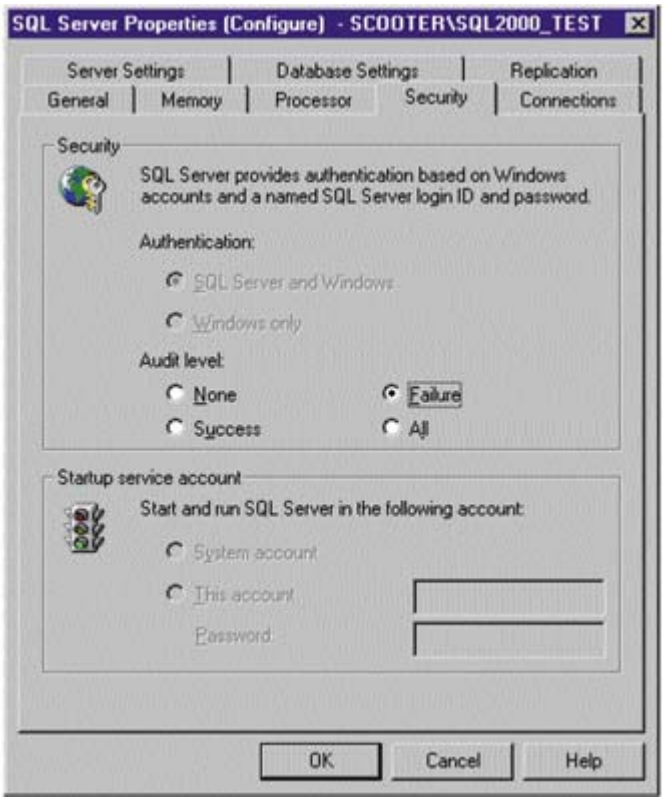


Figure 2.18: The SQL Server Properties Security tab.

Under the Auditing section, selecting All audits both successful and failed login attempts. You can record attempted user accesses as well as other SQL Server log information, and enable auditing for both security modes. You can also record information on both trusted and distrusted connections. Log records for these events appear in the Windows application log, the SQL Server error log, or both, depending on how you configure logging for your SQL Server. If you select this option, you must stop and restart the server to enable auditing.

In the Startup Service Account section, selecting “System account” specifies that the instance of SQL Server service account is the built-in local system administrator account.

In the Startup Service Account section, selecting “This account” specifies that the SQL Server service account is a Microsoft Windows NT 4.0 or Windows 2000 domain account. This field is enabled only if you are using a valid Windows NT 4.0 or Windows 2000 administrator account on the computer on which the registered instance of SQL Server is running.

In the Startup Service Account section, Password specifies the password for the Windows NT 4.0 or Windows 2000 domain account. This field is enabled only if you are using a valid Windows NT 4.0 or Windows 2000 administrator account on the computer on which the registered instance of SQL Server is running.

The Configured Values option button allows you to configure the options for the server. The Running Value option button allows you to view the current running values for the options on this tab. These values are read-only.

The Connections Tab

On the Connections tab (see Figure 2.19), the “Maximum concurrent user connections” setting specifies the maximum concurrent user connections. Entering 0 means there can be an unlimited number of concurrent user connections.

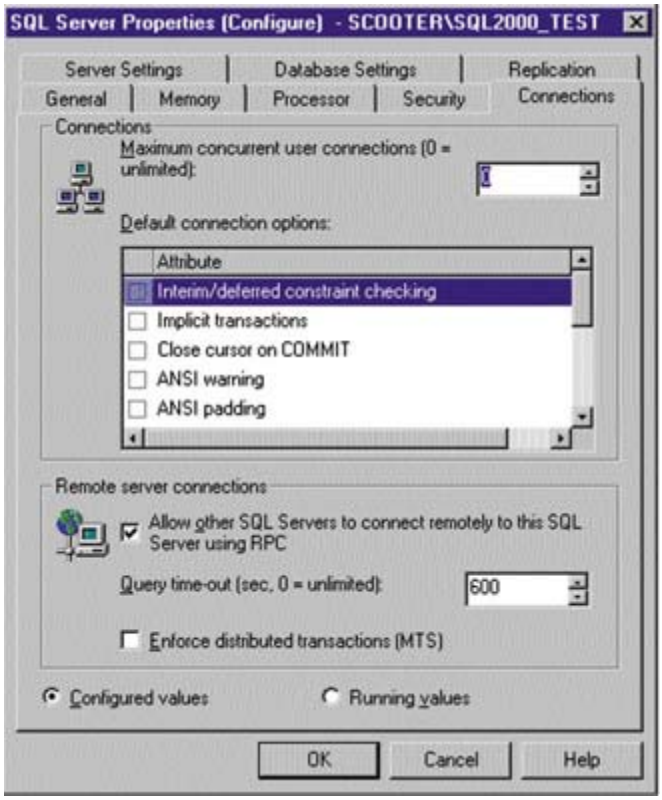


Figure 2.19: The SQL Server Properties Connections tab.

“Default connection options” allows you to specify the default connection options for the selected server.

In the Remote Server Connections section, checking the "Allow other SQL Servers to connect remotely to this SQL Server using RPC" selection will do as specified—it will allow other SQL Servers to connect remotely to this SQL Server using remote procedure calls. You might enable feature this in a multiserver network.

In the Remote Server Connections section, the "Query time-out (seconds)" option specifies the number of seconds that must elapse during a remote query before the query times out. Specifying 0 means that an unlimited amount of time can elapse.

In the Remote Server Connections section, the "Enforce distributed transactions (MTS)" selection protects a server-to-server procedure by using MS DTC to coordinate distributed transactions.

The Configured Values option button allows you to configure the options for the server. The Running Value option button allows you to view the current running values for the options on this tab. These values are read-only.

The Server Settings Tab

The settings for the server behavior are logically grouped together on the Server Settings tab (see Figure 2.20).

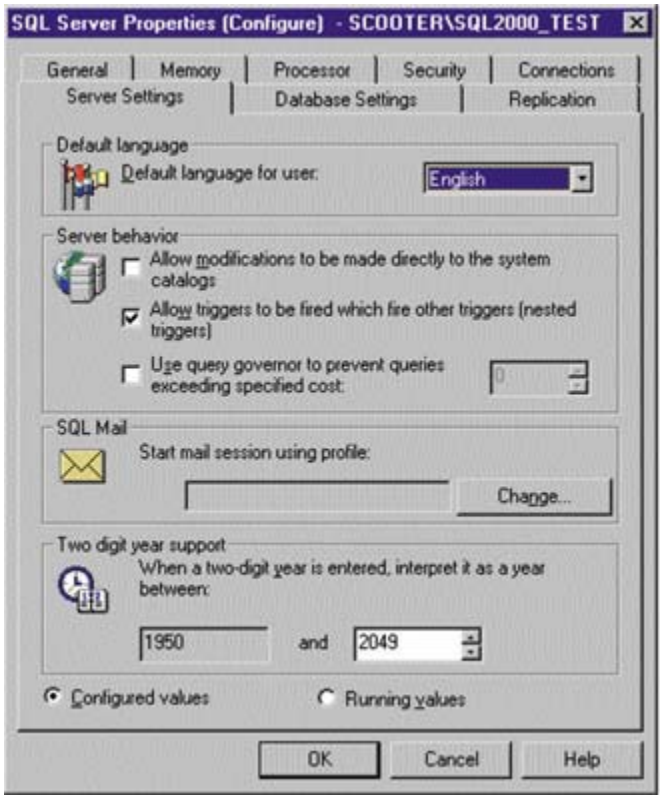


Figure 2.20: The SQL Server Properties Server Settings tab.

The “Default language for user” combo box allows you to specify the default language for server messages.

The “Allow modifications to be made directly to the system catalogs” check box allows modifications to be made directly to the system catalogs.

The “Allow triggers to be fired which fire other triggers (nested triggers)” check box allows nested triggers to be fired.

The “Use query governor to prevent queries exceeding specified cost” check box enables the query governor. This option is especially useful if users are performing ad hoc queries. It can help prevent them from creating queries that use too much CPU time. There is also a box that allows you to specify a query cost.

On this tab, you can specify a mail login name as well. This name will be used for mail sent from SQL Server.

The “When a two-digit year is entered, interpret as a year between” boxes are provided to allow you to specify how an instance of SQL Server interprets two-digit years. To change the time span, type the ending year. The default time span is 1950 to 2049. The beginning date is January 1, 1950, and the ending date is December 31, 2049. The number 99 is interpreted as 1999, and 01 is interpreted as 2001. The rule is that years less than or equal to the last two digits of the cutoff year are in the same century as that of the cutoff year. Years greater than the last two digits of the cutoff year are in the century previous to that of the cutoff year. Four-digit years are not affected by this option. If you want SQL Server to use the same two-digit cutoff year as the client, select 2030.

The Configured Values option button allows you to configure the options for the server. The Running Values option button allows you to view the current running values for the options on this tab. These values are read-only.

The Database Settings Tab

You can set the default index fill factor manually on the Database Settings tab (see Figure 2.21). The index fill factor determines how full Microsoft SQL Server makes each page when it creates a new index using existing data. When this option is cleared, SQL Server selects the optimal setting for performance:

- **Wait indefinitely** Specify that DB-Library must wait indefinitely for the instance of SQL Server to respond.
- **Try once then quit** Specify that DB-Library must try once to connect to an instance of SQL Server and then time out.
- **Try for minute(s)** Specify the time, in minutes, that DB-Library must try to connect to an instance of SQL Server before timing out.

- **Default backup media retention (days)** Set a systemwide default for the length of time to retain each backup medium after the backup has been used for a database or transaction log backup.
- **Recovery interval (Min)** Set the maximum number of minutes per database that SQL Server needs in order to complete its recovery procedures. The default is 0 minutes per database, which is the autoconfiguration for fast recovery.
- **Default data directory** Specify the default directory used for data files when new databases are created in SQL Server. Click the browse (…) button to search for an existing data directory.
- **Default log directory** Specify the default directory used for log files when new databases are created in SQL Server. Click the browse (…) button to search for an existing log directory.

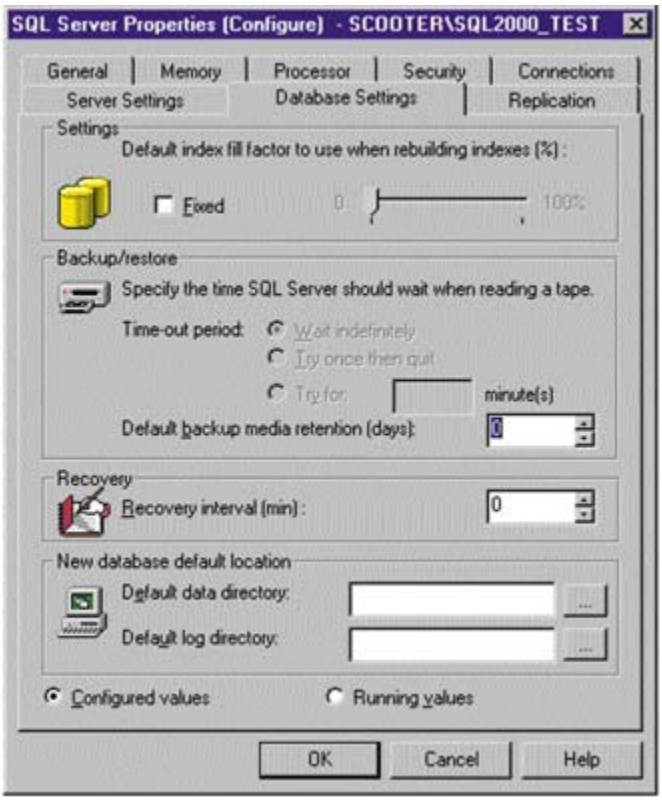


Figure 2.21: The SQL Server Properties Database Settings tab.

The Configured Values option button allows you to configure the options for the server. The Running Values option button allows you to view the current running values for the options on this tab. These values are read-only.

The Replication Tab

The Replication tab allows you to configure replication settings for SQL Server 2000 (see Figure 2.22).



Figure 2.22: The SQL Server Properties Replication tab.

Server Network Utility

SQL Server Client Network Utility is a graphical tool that allows you to specify the network protocols used by the server. It also allows you to display the DB-Library version currently installed on the system and set defaults for DB-Library options.

The General Tab

The General tab (see Figure 2.23) allows you to enable and disable the installed protocols on a server. It also allows you to configure the server to force protocol encryption and enable and set proxy settings.

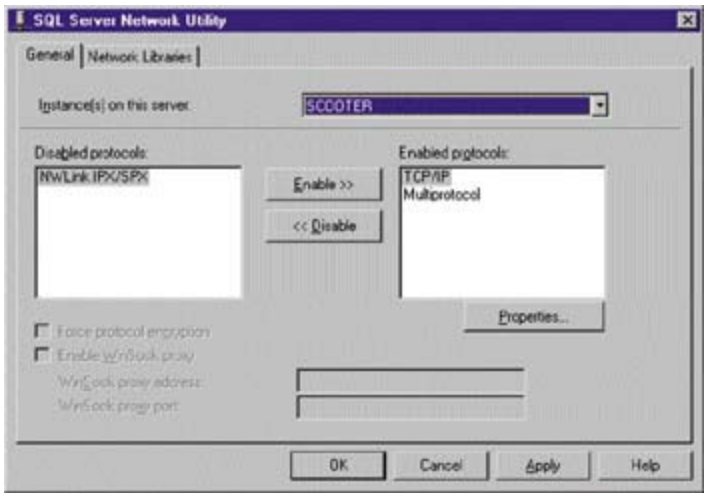


Figure 2.23: The SQL Server Network Utility General tab

You would enable only the protocols being used on the clients. As part of the planning stages for a new installation, you should select a protocol to use and be consistent. In some cases, this is not practical (clients need to access other systems and clients are already set up). In these cases, you should enable the needed protocols.

Tip Enable only protocols you need to use, because enabling unused protocols adds system overhead and reduces performance.

The Network Libraries Tab

The Network Libraries tab allows you to view the installed network libraries on the server (see Figure 2.24).

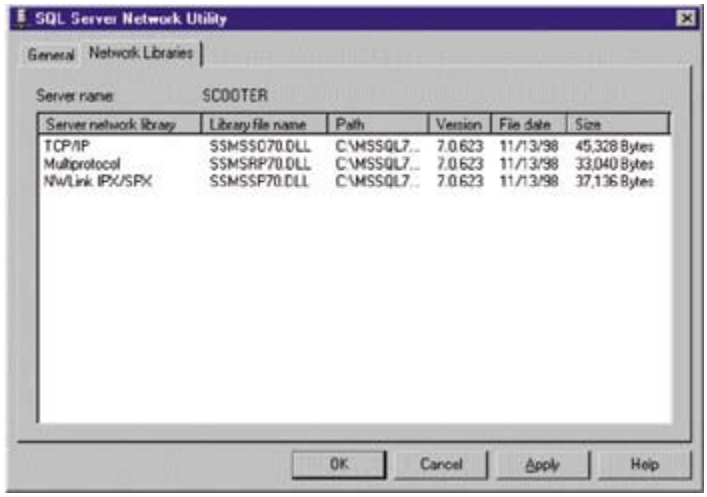


Figure 2.24: The SQL Server Network Utility Network Libraries tab.

Client Network Utility

SQL Server Client Network Utility is a graphical tool that allows you to:

- Create network protocol connections to specified servers, and change the default network protocol.
- Display information about the network libraries currently installed on the system.
- Display the DB-Library version currently installed on the system, and set defaults for DB-Library options. The General tab allows you to view and configure the installed network protocols on the client (see Figure 2.25).

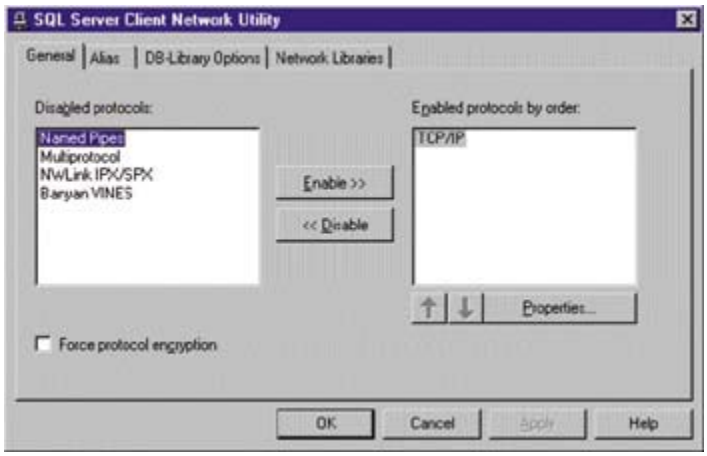


Figure 2.25: The SQL Server Client Network Utility General tab.

The Alias tab allows you to configure alias names for servers based on IP address or server name (see Figure 2.26).

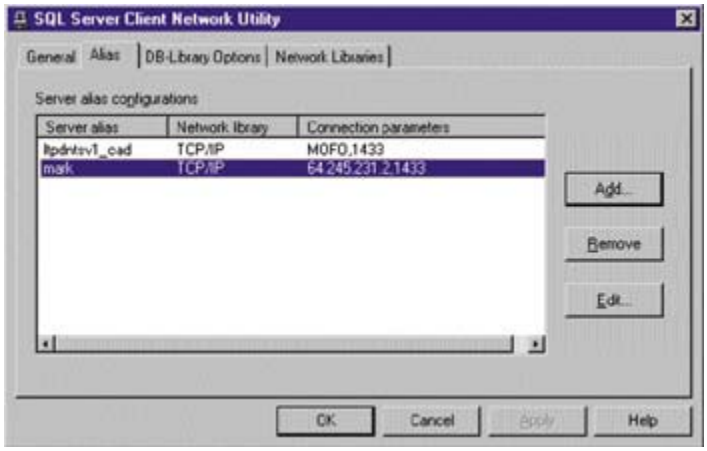


Figure 2.26: The SQL Server Client Network Utility Alias tab.

The DB-Library tab allows you to configure options for the DB library (see Figure 2.27). Additionally, the version information about the installed DB library is available on this tab.

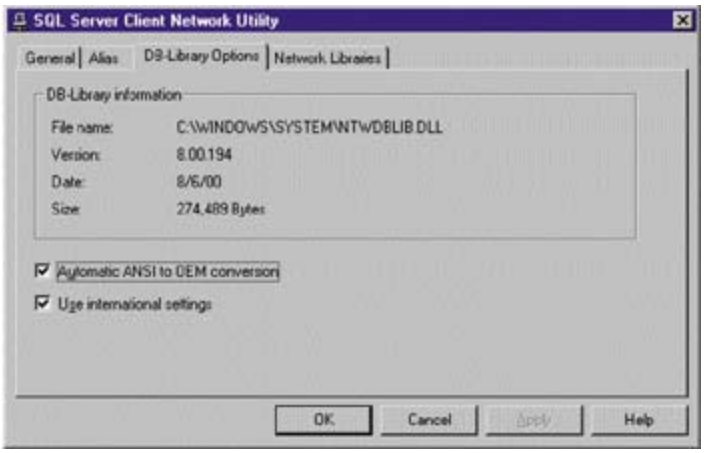


Figure 2.27: The SQL Server Client Network Utility DB-Library tab.

All of the network libraries and their versions are displayed on the Network Libraries tab (see Figure 2.28). This information is especially useful for troubleshooting purposes.

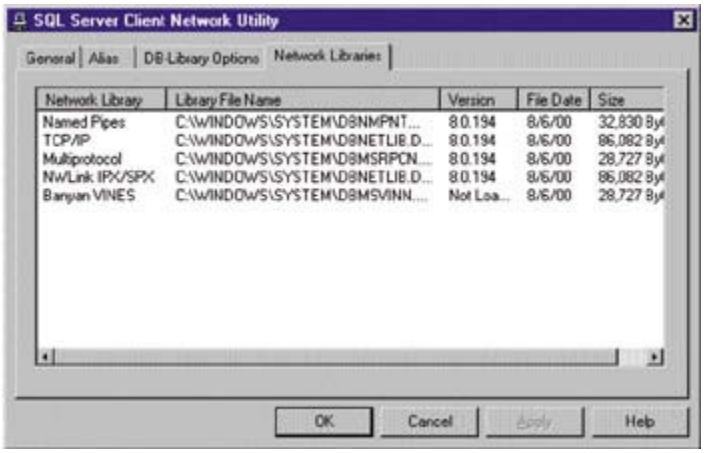


Figure 2.28: The SQL Server Client Network Utility Network Libraries tab.

SQL Server Agent

SQL Server agent is installed with SQL Server. The property pages for SQL Server Agent are outlined here:

The General tab allows you to configure the service account for SQL Agent (see Figure 2.29). It can be a different account from the one under which SQL Server runs or the same account. You can configure SQL Server Agent to run under the system account, but this is not recommended, because in many cases SQL Server Agent needs to communicate with multiple servers.

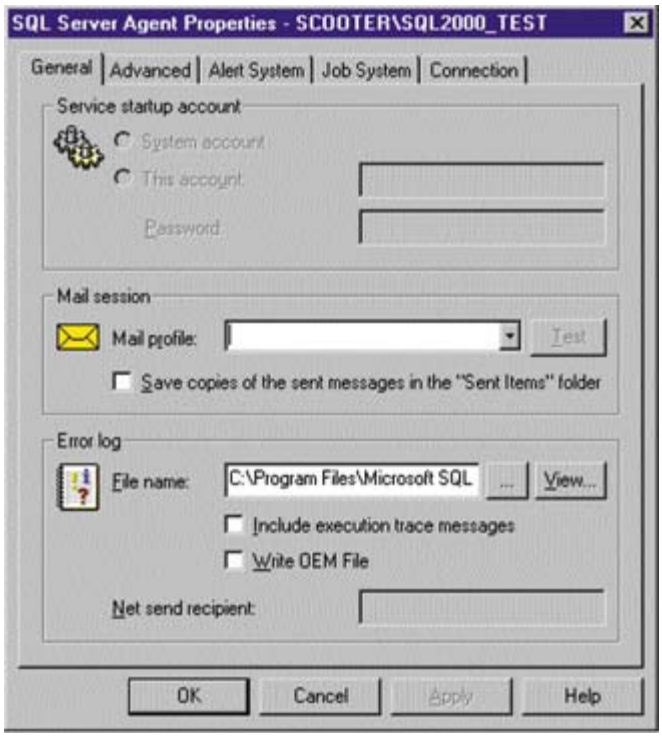


Figure 2.29: The SQL Server Agent Properties General tab.

Mail profile is the MAPI mail profile to use.

The “Error log” section allows you to configure an error log file for SQL Agent. The error log can be configured to include execution trace messages—useful for troubleshooting jobs.

The Advanced tab allows you to configure SQL Agent to restart if it fails or if SQL Server fails (see Figure 2.30). Additionally, event forwarding and what SQL Agent considers idle time can be configured here. It is useful to tune the agent to your server.

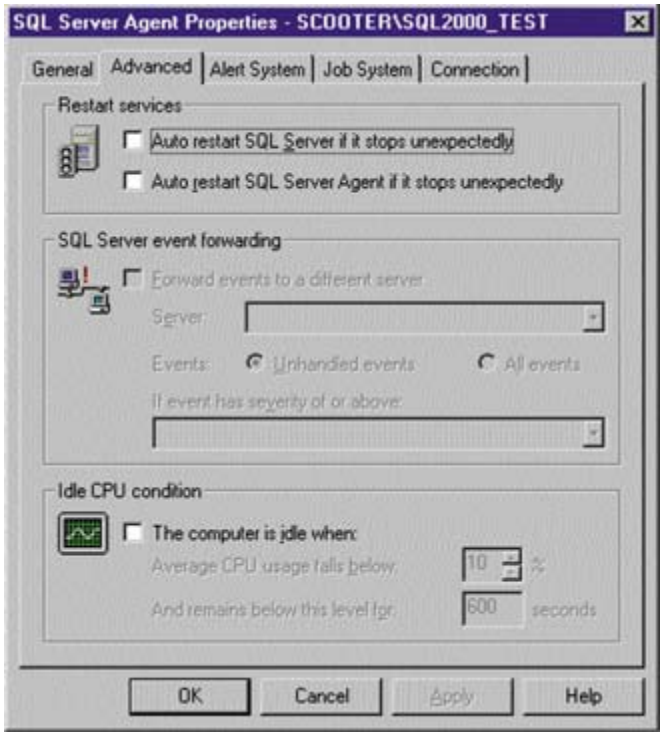


Figure 2.30: The SQL Server Agent Properties Advanced tab.

Alerting profiles can be set up on the Alert System tab (see Figure 2.31). This option configures where alerts will be sent via e-mail. It's important that mail be configured in order for this option to function properly. Alerts can be sent to multiple e-mail addresses and copied to multiple e-mail addresses.

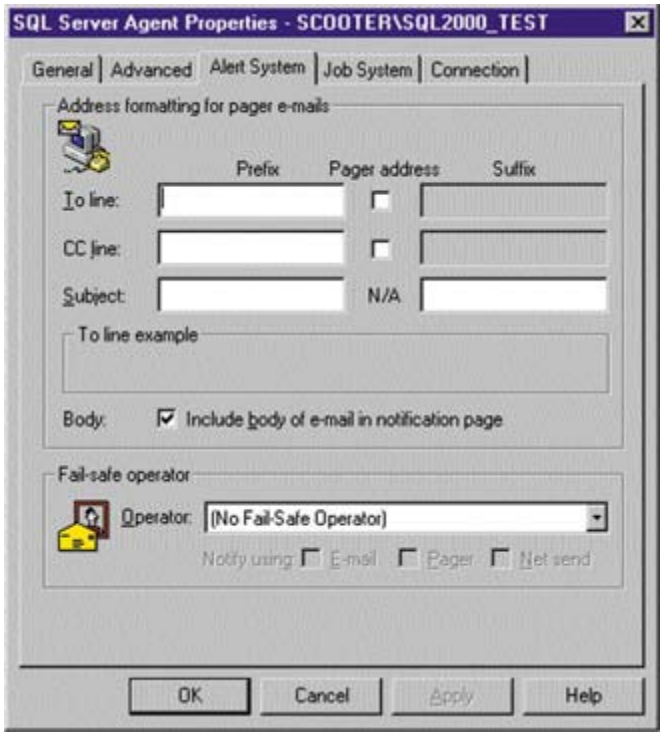


Figure 2.31: The SQL Server Agent Properties Alert System tab.

The Job System tab allows parameters to be configured with relation to the jobs SQL Agent runs, such as replication (see Figure 2.32). The job history log size can be limited by rows as well as per job.

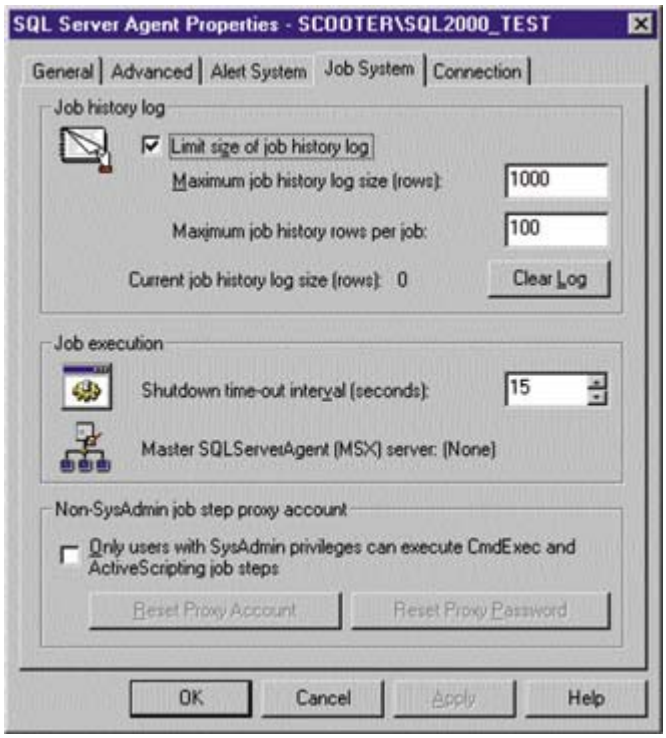


Figure 2.32: The SQL Server Agent Properties Job System tab.

The Connection tab allows you to configure the connection to SQL Server (see Figure 2.33). You can set the account to use as well as the login time-out. Additionally, you can configure the local host server alias.

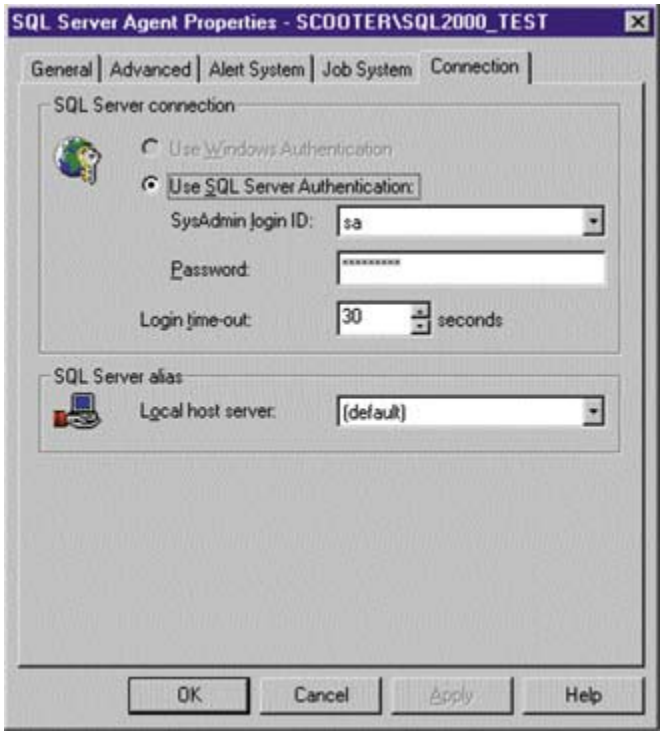


Figure 2.33: The SQL Server Agent Properties Connection tab.

SQL Mail

The SQL Mail Properties page allows you to pick a MAPI profile name (see Figure 2.34). You must first configure a MAPI mail profile prior to installing SQL Mail.

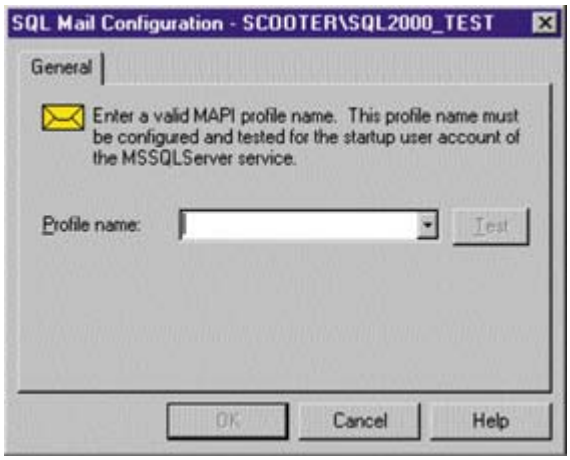


Figure 2.34: The SQL Mail Configuration screen.

Summary

In this chapter, we have covered the installation of SQL Server 2000 as well as the English Query and Analysis services.

SQL Server is much easier to install than easier versions, which can lead to installing SQL Server without spending enough time up front analyzing the system requirements and architecture. One of the key points you should understand is that Microsoft SQL Server is very powerful when it is properly installed.

SQL Server offers many flexible installation options, from installing only the client tools to installing the server with the English Query and Analysis tools. The enhancements to the tools, the ability to have multiple instances, and the ease of installation all make SQL Server 2000 a great database system. You were also exposed to common upgrade scenarios, upgrading from SQL Server 6.5 and SQL Server 7.0, and upgrading requirements and issues.

We covered the various configuration screens for SQL Server 2000. We have explored the options on the configuration screens; many of these are covered in greater detail in later chapters. We explored clustering and talked about some of the installation pitfalls and issues surrounding use of clustering. And finally, we discussed the importance of planning your installation. Proper planning will help ensure a successful installation and migration to SQL Server 2000.

FAQs

Q. Can I run SQL Server 2000 and SQL Server 7.0 on the same machine?

A. Yes. They can run concurrently. The SQL Server 7.0 instance becomes the default instance.

Q. Can I have more than one default instance for SQL Server 2000?

A. No. Only one instance can be the default. All others are named instances.

Q. Can I make the SQL Server 2000 instance the default and the SQL Server 7.0 a named instance?

A. No, the SQL Server 7.0 instance needs to be the default.

Q. How can I change the password for SQL Server Service?

A. The easiest way is to use Enterprise Manager to change the password on SQL Server. Additionally, you need to change the password on the domain. It is important to keep the passwords synchronized.

Chapter 3: SQL Server Scalability and Availability

Introduction

Every technology professional has faced issues concerning the availability and scalability of his or her hardware and software solutions. Continued commitment to making SQL Server a player in all markets, from handheld and portable applications to enterprise-scale database solutions, is evident in SQL Server 2000. This latest release of SQL Server provides support for up to 64GB of RAM and 32 processors, with Enterprise Edition running on Windows 2000 Datacenter Server. At the other end of the spectrum, SQL Server now boasts support for database development on Windows CE Edition. No other database system provides this level of platform scalability with a programming model that is this consistent, thus allowing SQL Server application developers to leverage their skills on any platform.

Improved platform support is complemented with scalability and availability enhancements, including distributed partitioned views through federated database servers. This scale-out approach allows database solutions to be distributed across numerous independent servers following a share-nothing approach to clustering while enhancing both scalability and availability. Indexed views, another addition to SQL Server 2000, will reduce the strain on many reporting-intensive database solutions such as OLAP systems.

SQL Server 2000 offers extended support for fail-over clustering, providing near 100 percent availability. The setup of SQL Server fail-over clustering is simplified by automatic detection of Windows 2000 cluster service. Enhanced cluster management improves usability. The new Log Shipping feature supports configuring warm standby servers for fail-over or read-only server roles.

Whether you are building a remote, distributed database application or a massive data warehouse, SQL Server 2000 offers the scalability and availability necessary to meet those goals. This chapter discusses these enhancements and additions to SQL Server 2000 as well as assists you in configuring your solutions to take advantage of these enhanced technologies.

Scaling Up vs. Scaling Out

In the past, when you needed to increase the performance of your database applications, you would probably think of *scaling-up* techniques such as purchasing a replacement database server with more horsepower or upgrading your old hardware. SQL Server 2000 and Windows 2000 enable you to *scale out* your databases and applications across multiple servers using cluster services or federated database servers, rather than simply scaling up.

SQL Server is a multithreaded application that supports scaling up with symmetric multiprocessor (SMP) technology. Query performance can be greatly enhanced by increasing the number of processors in the server, since queries can be processed in parallel. However, scaling up in this manner requires very specific and expensive hardware such as Pentium XEON processor technology in quad or higher processor configurations.

Typically, SMP does not support high availability; the operating system and its data structures form a single point of failure (SPOF) because of their "share everything" (e.g., memory, storage systems, and the like) nature. Certain failures in the hardware, such as a processor or memory module failure, or in the operating system itself are likely to cause the entire system to crash, making it unavailable to your applications. However, in other cases, such as a software component failure, the system can reboot after discarding the faulty component. SMP provides a single system image due to its coherent shared memory and the fact that it is running a single copy of the operating system. The tight connection of processors to the rest of the system usually makes the lifetime of a system design based on the SMP model relatively short, compared with networked systems. SMP is, by definition, a board-level technology, so to upgrade, you'll need to replace the system board (which usually translates to having to buy a whole new system). That is, a given motherboard design can accommodate a limited number of processor generations (two or three). By *processor generation* , we mean major evolution (e.g., from Pentium II to Pentium III), not simple increases in operating frequency

SQL Server 2000 scales up extremely well. Enterprise Server running on Windows 2000 Server supports up to two processors (four when upgrading from NT Enterprise) and 4GB RAM. Windows 2000 Advanced Server can scale up to four processors (eight when upgrading from NT Enterprise, and Windows 2000 Datacenter Server supports up to 16 processors—32 through OEMs). Both Windows 2000 Advanced Server and Datacenter Server support up to 64GB of RAM on 64-bit processors such as DEC Alpha and Intel Pentium II XEON systems.

Scaling out allows you to install your database (or portions of it) on multiple servers as a group, commonly called a *server farm,* or cluster. Each of the member servers (nodes) in the cluster maintains a synchronized copy of the data, on either completely separate storage systems (share nothing) or by storing a single copy of the data on a shared disk. Should one of the servers go down, another server can seamlessly take over for it via

automatic or manual fail-over, so that the database or application will continue to be available.

Scale-out in SQL 2k is implemented via support for two- and four-way clusters using Microsoft Cluster Service or federated database servers implementing share-nothing technology with database partitioning.

In addition to high availability, application-level, *share nothing* , clustering can provide an increase in system throughput, although system throughput of a cluster is limited by the characteristics of the running application. Microsoft's new Application Server product allows previously "single-server only" applications to be scaled out to allow sharing of their workload across multiple "federated" servers. According to recent Transaction Processing Council (TCP) benchmarks, SQL Server applications running on Microsoft COM+ clusters, with proper tuning, scale extremely well compared with SMP systems, because interactions (due to data sharing among nodes) can be limited by the proper tuning of both the database server and the applications. SQL Server 2000 has been adapted to exploit Application Server cluster architecture for parallel query processing. The system speed can be improved in the share-nothing cluster model, provided that applications have been adapted to benefit from parallelism.

System Architectures and Database Scalability

An important dimension of planning your system architecture for database scalability is the relationship between mass storage and processing/memory elements (see Figure 3.1).

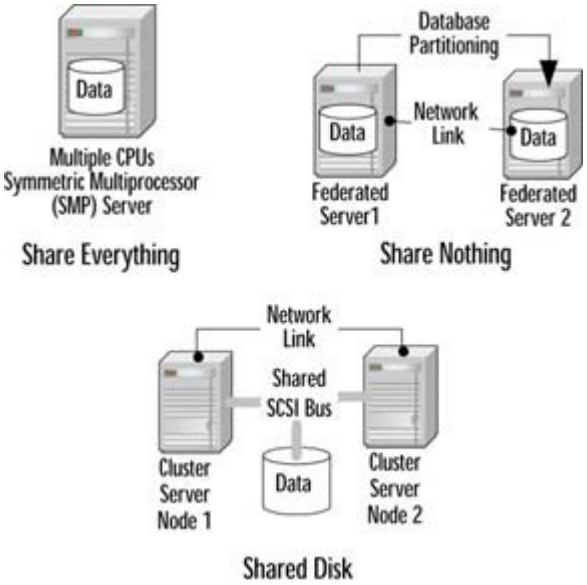


Figure 3.1: System architecture affects database scalability.

- **Share everything** A single server containing multiple processors, which share all memory and all mass-storage devices . The processors share the workload as well, so application performance can be enhanced dramatically by increasing the

number of processors. The drawback of this architecture is its inherent lack of fault tolerance—that is, if any major component (such as memory or a processor) fails, the entire system fails. *SMP systems follow this architectural model. SQL Server supports SMP and can be scaled up to 32 processors on Windows 2000 Datacenter Server.*

- **Share nothing** Two or more server nodes composed of either similar or dissimilar hardware—processor(s), memory, and storage devices—are linked together through an interconnect (Fast Ethernet, FDDI, and so on). Nodes share no physical resources; all information exchanges are accomplished via messaging. In addition, overall system fault tolerance can be increased since at least a portion of the data might still be available if one system fails. *Partitioning of a SQL Server database across multiple federated servers is an example of this type of architecture, as is Microsoft Application Server application-level clustering.*
- **Shared disks** A cluster of two or more servers containing separate processors and memory are linked through a storage channel such as a shared SCSI bus. Each of the nodes can access any of the drives on the shared bus, but the exchange of information between computing nodes is accomplished via messaging. This is a highly available solution since one server can go down but applications can continue to access the data on the shared disk via the other secondary server. Performance is generally not improved in this model because overhead is involved in sharing the disk channel. *Microsoft Cluster Service and SQL Server Fail-over Clustering utilize this model.*

TPC Benchmarks

The *Transaction Processing Council (TPC)* maintains current performance benchmark data for a variety of platforms and applications, including SQL Server. For example, you might want to search its Web site to see if the council has specific benchmark data on that new database server your company is considering purchasing.

Industry-standard benchmarks, defined by the TPC, provide a recognized way for companies to assess both competitive performance and price/performance for two critical aspects of business computing:

- **TPC-C** Benchmarks for OLTP throughput are most commonly used for assessing application performance. These benchmarks reflect the real-world performance of OLTP applications such as e-commerce, manufacturing execution systems, travel reservations terminals, stock trading, bank tellers, and the like. Throughput for the TPC-C benchmark is measured in sustained transactions per minute (tpmC) and total cost of ownership (TCO) of a particular system in dollars divided by transactions per minute (price/tpmC).
- **TPC-H** Benchmarks for decision support are used specifically for gauging database query performance. Server-intensive applications requiring manipulation

of large amounts of data (such as data warehouse queries or OLAP cube generation) are measured by TPC-H benchmarks. Some examples are analysis of census results or sales trends over several years. The TPC-H benchmark is called the *query-per-hour performance metric* and is expressed as queries per hour categorized by database size (QphH @size). The TPC-H price/performance metric is expressed as $/QphH @size.

In October 2000, Compaq published TPC-C Benchmark results for SQL Server 2000 that consistently topped previous transaction processing throughput (505,302 tpmC) and price/performance (20.68$/tpmC) records set by Oracle and DB2. The benchmark was generated by a Compaq Proliant 8500 Cluster consisting of 24 servers, each with eight 700MHz CPUs and 8GB RAM. Compaq and SQL Server 2000 also topped the list for TPC-H on 100GB databases with 1699QphH and 161 US$/QphH on a nonclustered Proliant 8000 server with eight 700MHz CPUs and 4GB RAM.

You can view the current top 10 TPC-C benchmarks at www.tpc.org/New_Result/TPCC_Results.html and the top 10 TPC-H results at www.tpc.org/new_result/h-ttperf.idc .

SQL Server Fail-Over Clustering

According to a 1999 survey of large corporations, more than half of all servers are required to be up 100 percent of the time on a 24 x 7 basis. This percentage is expected to grow with the increase in demand for application availability on the Web (where the stores are "always open"). Another survey reported that the average cost of downtime of business-critical applications can exceed $10,000 an hour, excluding additional losses caused by resulting customer dissatisfaction. Building a completely fault-tolerant SQL Server system might seem initially very expensive, but as you can see, the cost of a single unplanned outage could easily outweigh it.

SQL Server 2000 Fail-over Clustering allows you to configure two servers, or up to four in the case of Windows 2000 Datacenter Server, to function as a single "virtual server" for a specific database instance. In the event of one server's failure, the other server(s) takes over transparently and continues to provide users with access to the database instance. Another benefit to fail-over clustering is that one server can be temporarily removed from the cluster for maintenance or hardware upgrades, without database downtime.

SQL 2000 Fail-Over Clustering Architecture

Fail-over clustering supports multiple SQL Server instances. Multiple-instance support makes it easier to build, install, and configure SQL Server virtual servers in a fail-over cluster environment. Each instance has its own set of system and user databases. Multiple-instance support allows you to isolate work environments (for example, testing from production) or volatile application environments and provide different system administrators for each instance of SQL Server on the same computer.

You can run multiple instances of SQL Server on the virtual server of a SQL Server fail-over cluster. When you install an instance of SQL Server 2000, you can specify up to four computers that make up the cluster for that instance. The cluster looks like a single computer to applications connecting to that instance of SQL Server. When applications connect to the cluster, they specify the virtual server name of the cluster and the instance name in the form *virtual_servername\instancename* .

Planning for SQL Server Clustering

Figure 3.2 shows a typical two-node server environment for SQL Server Fail-over Clustering. The two servers in this example are connected to each other by two separate network connections; one connect the servers to the external network and one is connected to an internal subnet "heartbeat" linking the two servers so that each can monitor the status of the other. Both servers have internal boot disks on which their operating systems and SQL Server executables are stored. The servers are also connected via a shared SCSI channel to an external hard disk array, which is where the database is physically stored. A "quorum disk" partition containing the cluster's configuration and synchronization database checkpoints and log files is also stored on this shared disk.

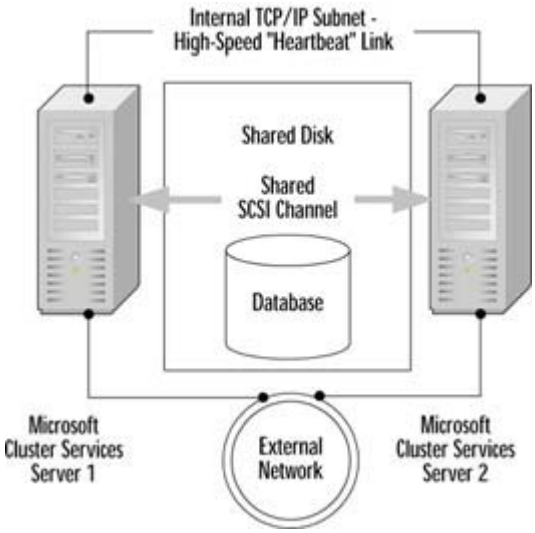


Figure 3.2: Planning your SQL Server 2000 Fail-Over Clustering environment.

The primary server controls the shared disk channel. The primary server is usually the server that booted first. When the primary server fails, the secondary server takes control of the channel automatically. Cluster services "keep alive" client connections and redirect their requests to the secondary server. Also on fail-over, the cluster's registry data is copied from the primary node to the secondary node.

Clustering Capabilities and Requirements

Clustering is a very powerful means of achieving fault tolerance, and its benefits are numerous. It is also relatively expensive to implement. Identical, specialized hardware and software must be installed on each of the member servers in a cluster. The member servers must be able to communicate with each other at all times. When a communication link is lost, the surviving server must be able to seamlessly carry on the work of its failed counterpart.

Hardware Compatibility

Microsoft Cluster Service (MSCS) requires that the member servers use a shared disk channel. This is typically a specialized SCSI controller with hardware RAID to protect the data on your cluster storage. A test is performed during configuration that will disallow the configuration of MSCS unless the system has compatible hardware. Prior to MSCS installation, you should check the Microsoft Windows Hardware Compatibility List site for a list of required clustering-compatible hardware. That Web address is www.microsoft.com/hcl/default.asp .

To further increase the availability of network resources and to prevent the loss of data:

- Try to keep replacement hardware in stock at your site. The hardware devices most likely to fail are hot-swap drives, RAID controllers, and network cards.

Keeping spares around avoids having the server down while you are waiting for a part to be shipped from a dealer after the failure occurs.

- Provide backup power for individual computers as well as for your network infrastructure. If you can afford it, a power generator or "mainframe-sized" battery backup can keep all your PC servers running for hours or even days without power. The use of individual UPS protection ensures 5 to 20 minutes in which the system can shut down in an orderly fashion before power failure can cause loss or corruption of data. In addition, a conditioned power source can prolong the life of both hard drives and memory, which are very sensitive to power surges and brownouts.

Software Compatibility

MSCS can be run on several different operating systems, but all computers in an individual cluster must have the same operating system installed.

Microsoft Cluster Service

MSCS can be installed on Microsoft Windows NT Server 4.0 Enterprise Edition, Windows 2000 Advanced Server, or Windows 2000 Datacenter Server. MSCS is a set of optional services that can be installed from the Windows 2000 Advanced/Datacenter Server installation CD, but it comes built into Windows NT Enterprise Edition.

A SQL Server 2000 Fail-over Cluster can consist of two nodes on all of these operating systems, with the exception of Windows 2000 Datacenter Server, which supports four-node clustering. Microsoft strongly recommends using TCP/IP instead of named for communications between clustered servers; SQL Server Fail-over Cluster requires you to use TCP/IP.

System Area Networks

The computers in a cluster must be connected via a *system area network (SAN).* This should be 100Mbps Fast Ethernet at a minimum but preferably fiber optic or Gigabit Ethernet. In a two-server cluster, use a short cross-over cable (TX pairs wired to corresponding RX pairs on the other end) for best results. For a four-node cluster, use a small, high-performance switch not connected to the rest of the network.

All network interfaces used on all nodes in a server cluster must be on the same network. All cluster nodes must be on the same subnet.

SANs are not TCP/IP based 100Mbps and are not required for heartbeat configuration. SANs are ultra-high-speed, hardware-controlled communications networks between servers that are possible configurations but not required. A heartbeat link can be implemented using standard switched 100Mbps Ethernet. Don't confuse SANs with standard private network links.

Shared Disks

The cluster requires that all shared disks and the quorum disk be physically connected to a shared bus via appropriate cables. All disks must be visible to all nodes in the host adapter setup software (refer to your manufacturer's documentation). We strongly recommend that you configure them as a fault-tolerant RAID array (such as RAID1 or RAID5) rather than as a striped set without parity.

The cluster's shared disks must be formatted using the NT File System (NTFS) for security and added stability. Shared disks *must* be configured as basic disks; Windows 2000 does not support the use of dynamic disks for cluster storage. In addition, you cannot use software RAID to protect the data on the cluster storage; for example, you cannot use a software fault-tolerant set or volume set. Additionally, the Encrypting File System, the Remote Storage utility, and mounted volumes are not supported by MSCS clusters.

Log Shipping

If you do not have the luxury of being able to purchase perfectly matched pairs of cluster servers for fault tolerance, the log-shipping feature can provide you with an inexpensive standby server. Log shipping does not allow for automatic fail-over but also does not require identical hardware. See the Log Shipping section later in this chapter for further details.

Implementing Fail-Over Clustering

Prior to installing the Cluster Service software, ensure that you have properly installed and configured your server operating system, at least two separate network cards, and both individual and shared hard disks on each node in the cluster. All server nodes should be set up as either all as member servers or all domain controllers within the same domain (don't mix them). You will also need to predefine the following:

- A valid NetBIOS name for the cluster "virtual server" (it cannot be named the same as any member server or any other server on your network).
- The Cluster Service domain login account.
- On each server, you must have two unique static IP addresses—one for the heartbeat link adapter and one for the external network adapter. The private IP addresses must be on an IP subnet separate from the external network. In addition, you must define at least one unique static IP address for the cluster itself to use.

Setting Up Network Adapters

You must use at least two separate network adapters in each cluster node—one for the private connection to the other server(s) in the cluster and one for the external connection to the public network. The private network card will be used for the "heartbeat link"—communication between the nodes to determine their up status as well as for cluster

management. Physically connect the private network using either a direct "cross-over" cable (if you have only two nodes) or a high-speed switch. Ensure that TCP/IP has been configured on each card, with the proper IP address and subnet masking applied to each. Check communication between servers using the PING command-line utility.

Setting Up Shared Disks

Set up the shared disk array using the manufacturer-specific software that came with your disk adapters. Run any available read/write tests to ensure that your cable connections and termination are set up correctly. Test that each server can take control of the shared drive array by powering it up with none of the others turned on. Install the operating system on each server's local drive, but leave all but one server powered off to prepare the shared hard drives and to install Microsoft Cluster Services.

Warning To prevent shared disk corruption, do not power on any of the other member servers until disk partitioning and formatting are completed successfully on the first node.

Using Disk Manager on NT (or the Computer Management | Disk Management Utility in Windows 2000), create a small (50MB minimum, 500MB preferred) partition on the shared array to use as the quorum disk for cluster management data storage. Create your other shared disk partitions, keeping in mind your SQL Server database design. For example, you might want to have a separate partition for transaction log backup or for the TempDB database.

Format the quorum disk and all data partitions with NTFS. Assign a drive letter to each, which you will need to know for cluster configuration. Power up your additional servers one at a time, allowing them to take control of the array, and ensure that each can read and write from the new partitions on the shared disk.

Setting Up MSCS on NT 4.0

The following steps should be used to install MSCS on each Windows NT 4.0 server in the cluster. Note that during Windows NT setup, you will be prompted to install Microsoft Internet Information Server 2.0 (IIS). It is recommended that you uncheck this option.

1. Install Windows NT 4.0, Enterprise Edition and Service Pack 6a (or later). Service Pack 3 or higher is required to install Microsoft Cluster Service.
2. Install Microsoft Internet Explorer 5 or later.
3. Install Microsoft Cluster Server (MSCS) from the product disc.
4. Using Cluster Administrator, you will need to manually create a Microsoft Distributed Transaction Coordinator (MS DTC) compatible resource group in which DTC setup can create its resources. Remember to include an IP address, network name, and clustered disk resource. MS DTC will be installed later by SQL Setup.

5. Install SQL Server 2000 Enterprise Edition.

Note Always reinstall the latest server service pack before installing your clustered applications, such as SQL Server itself. Furthermore, in a fail-over cluster configuration, SQL Server 2000 supports Windows NT 4.0 running only under an administrator account.

Setting Up MSCS on Windows 2000 (Advanced Server/Datacenter Server)

MSCS comes bundled with Windows 2000 Advanced Server, so there is no need to purchase additional software or licenses.

Installation

To install the cluster service component, use *one* of these steps:

- Check the Cluster Service box when prompted on the Windows 2000 Component screen during your initial installation of Windows 2000 Advanced Server.
- Go to Add/Remove Programs in Control Panel, select Add or Remove Windows Components, click the Components button, and then check the Cluster Service box (see Figure 3.3).

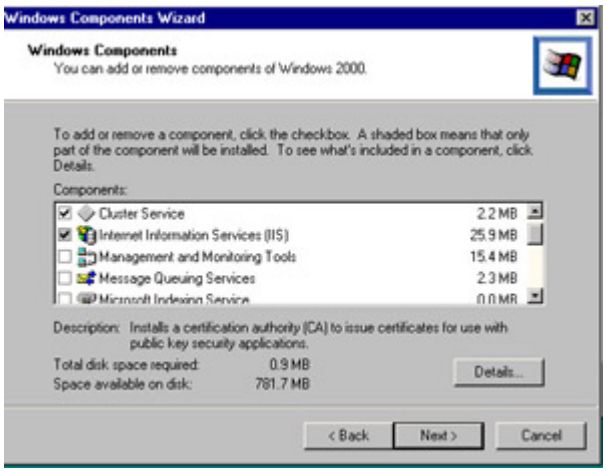


Figure 3.3: Select the Cluster Service component.

Configuration of Cluster Service

The Cluster Service Configuration Wizard will be displayed automatically at the end of Cluster Service installation. (You can also get to the wizard via Control Panel | Add/Remove Programs | Add/Remove Windows Components | Set Up Services.) Click the Configure button under the description of the Cluster Service.

MSCS management is handled using Cluster Administrator. To start Cluster Administrator, use *one* of these steps:

- Click Start, point to Programs, point to Administrative Tools, and then click Cluster Administrator.
- Run the executable %SystemRoot%\system32\cluster.exe. Enter the name of the cluster or the name/IP address of a member server in the Open Connection to Cluster dialog box.

Upgrading to SQL Fail-Over Clustering

You must use a domain account for all SQL Server services (SQL Server, SQL Server Agent, and the like), and all services within the clustered group. Administrator rights must be assigned to that account for all the servers in the cluster.

If you are upgrading a system with SQL Server 6.5 or 7.0 Fail-over Clustering currently installed, you must perform the following basic steps:

1. Be sure to apply the latest SQL Server service packs (at least SQL Server 6.5 SP5 or SQL Server 7.0 SP2) before beginning the upgrade.
2. Remove clustering of SQL Server 6.5 or 7.0 on both nodes using the Fail-over Cluster Wizard. SQL Server 6.5 or SQL Server 7.0 clusters cannot exist on the same computer as a SQL Server 2000 cluster.
3. Create a Cluster group for use with MS DTC.
4. Install SQL Server 2000 on the primary node first in the clustered configuration.

Warning It is strongly recommended that you do not run the Fail-over Cluster Wizard in SQL Server 6.5 or SQL Server 7.0 after SQL Server 2000 has been installed, because the results are unpredictable.

From SQL Server 7.0 Fail-Over Clustering

To upgrade from a previously installed SQL Server 7.0 fail-over cluster, upgrade one server at a time using the following steps:

Upgrading SERVER1:

1. Uncluster Microsoft SQL Server 7.0
2. Run SQL Server 2000 Setup to upgrade SQL Server 7.0 to SQL Server 2000.
3. During setup, create a SQL Server2000 clustered named instance.
4. Run the Copy Database Wizard to move all databases and related information from the newly upgraded default instance to the clustered named instance.
5. Uninstall the SQL Server 2000 default instance.

Upgrading SERVER2:

1. Uncluster SQL Server 7.0.
2. Upgrade SQL Server 7.0 to SQL Server 2000, and name it differently than both “default” and the instance name used for SERVER1.

3. Upgrade the SQL Server 2000 default instance to a clustered instance.

From SQL Server 6.5 Fail-Over Clustering

To upgrade from a previously installed SQL Server 6.5 active/passive fail-over cluster, upgrade both servers identically using these steps:

1. Uncluster Microsoft SQL Server 6.5.
2. Run SQL Server 2000 Setup to upgrade SQL Server 6.5 to a default SQL Server 2000 instance.
3. Migrate your data into SQL Server 2000 using the SQL Server Upgrade Wizard.
4. Uninstall SQL Server 6.5.
5. Run SQL Server 2000 Setup from the CD.

To upgrade from a previously installed SQL Server 6.5 active/active fail-over cluster, run the following steps on each server in turn:

Upgrading SERVER1:

1. Install a default (nonclustered) instance of Microsoft SQL Server 2000.
2. Run the SQL Server 2000 Convert Wizard. Migrate your data into SQL Server 2000.
3. Uninstall the SQL Server 6.5 instance.
4. Install a named SQL Server 2000 clustered instance.
5. Run the Database Copy Wizard. This migrates the 6.5 data to a named instance in SQL 2000.
6. Uninstall the default SQL Server 2000 instance.

Warning When upgrading to SQL Server 2000 from a SQL Server 6.5 or 7.0 active/active fail-over cluster (or any configuration in which SQL Server exists on the second node), you *must* first convert one side of the fail-over cluster to a named instance, when prompted. A name conflict will occur if both servers have "default" instances. In SQL Server 2000, only one default instance is allowed per fail-over cluster/server. We highly suggest renaming *all* instances to other than "default."

Upgrading SERVER2:

1. On SERVER2, install a default SQL Server 2000 instance.
2. Run the SQL Server 2000 Convert Wizard. Migrate your data into SQL Server 2000.
3. Uninstall the SQL Server 6.5 instance.
4. Upgrade the SQL Server default instance to a clustered instance.

Setting Up SQL Server 2000 Fail-Over Clustering

SQL Server 2000 requires MS DTC in the cluster for distributed queries and two-phase commit transactions as well as for particular replication functionality. You must run the Cluster Wizard after you install Microsoft Windows 2000 and configure your cluster. Run %SystemRoot%\system32\ comclust.exe on all nodes to configure MS DTC to run in clustered mode.

Once you have successfully installed and configured MSCS and MS DTC, run SQL Server 2000 Setup. SQL Server Setup automatically detects the existence of Cluster Service, installs a new instance of SQL Server binaries on the local disk of each computer in the cluster, and installs the system databases on the specified shared cluster disk. The binaries are installed in exactly the same path on each cluster node, so you must ensure that each node has a local drive letter in common with all the other nodes in the cluster.

To add additional servers to an existing cluster:

1. In the Welcome dialog box of the SQL Server Installation Wizard, click Next.
2. In the Computer Name dialog box, select Virtual Server and specify the virtual server to which you want to add a node. Click Next.
3. In the Installation Selection dialog box, select Advanced Options. Click Next.
4. In the Advanced Options dialog box, select Maintain a Virtual Server for Fail-over Clustering. Click Next.
5. In the Fail-over Clustering dialog box, click Next. *You do not need to enter an IP address* .
6. In the Cluster Management dialog box, select the server node to be added to the fail-over cluster and click Add. Click Next. Note that if the node is listed as unavailable, you must modify the disk resources in the virtual server's cluster group (by modifying your shared SCSI controller's configuration) so that the disk is available for the node you want to add to the SQL Server configuration.
7. In the Remote Information dialog box, for the remote cluster node enter login credentials that have administrator privileges on the remote node(s) of the cluster. Click Next.
8. In the Setup Complete dialog box, click Finish.

Warning The name you provide for the virtual SQL Server instance must be unique and must not be the same as any instance names used on the member servers. This is because SQL Server 2000 depends on distinct registry keys and service names within the cluster to smoothly continue after a fail-over.

To connect to a clustered instance of SQL Server 2000 running on a virtual server from a client, use the string *VIRTUAL_SERVER_NAME\instance-name.* You cannot use the name of the member server on which the SQL Server instance currently resides, because SQL Server does not listen on any IP address but the clustered virtual server's address.

Distributed Partitioned Views

Distributed partitioned views can be created to allow applications to transparently query data from one or more linked partner servers, thereby distributing the query load. SQL Server 7 had read-only partitioning capability only, whereas SQL Server 2000 gives you the ability to update the underlying tables of the partitioned views as well.

This scale-out method supports the growth of enterprise applications with SQL Server 2000. This method works best if the data can be separated based on related keys such as customer type, part number ranges, and geographic locales.

Note If a local or distributed partitioned view is not updateable, it can serve only as a read-only copy of the original table. An updateable partitioned view can exhibit all the capabilities of the original table.

Federated Servers

Federated servers are partner servers that allow the database workload to be divided, horizontally, throughout the network using a distributed partitioned view. Each federated server "owns" a copy of a unique portion of the entire data set.

How to Recover from Fail-Over Cluster Failure

Two main scenarios could cause a failure in the fail-over cluster:

- A faulty hardware device such as a network card or SCSI controller causes Node 1 (SERVER1) to crash.
- Cluster failure is caused by Node 1 being down or offline but not irretrievably broken. This could be caused by an operating system or application error (i.e. the "blue screen of death").

In Scenario 1, a hardware failure, failure is caused by a hardware failure in Node 1 of a two-node cluster. This hardware failure could be caused by, for example, the SCSI card or the network card failing. After Node 1 fails, the cluster fails over to Node 2. The following steps must be taken:

1. Node 1 must be evicted from MSCS. Go to Cluster Server Manager on Node 2 (SERVER2) and right-click the node to remove, then click Evict Node.
2. Run SQL Server Setup and remove Node 1. Install new hardware to replace the failed hardware in Node 1.
3. Install the operating system.
4. Install MSCS and join the existing cluster.
5. Run SQL Server Setup on Node 2 and add Node 1 back to the fail-over cluster.

In Scenario 2, a node gone offline, failure is caused by Node 1 being down or offline but not irretrievably broken. This could be caused by an operating system failure. Should Node 1 fail, the cluster fails over to Node 2. Do the following:

1. Run SQL Server Setup and remove Node 1.
2. Resolve the problem with Node 1.
3. Ensure that the MSCS cluster is working and that all nodes are online.
4. Run SQL Server Setup on Node 2 and add Node 1 back to the fail-over cluster.

Each of the partner servers can contain nonpartitioned data and the servers do not have to have identical hardware configurations, but the servers should be physically connected by a SAN with very high-speed, reliable network connections (Fast or Gigabit Ethernet) in order to avoid degraded performance.

Data Partitioning

The goal in *partitioning* or *portioning* data is to achieve optimal performance on queries; therefore, each federated server should maintain at least 80 percent of the data to be accessed by its "local" applications, whereas 20 percent or less of the data, used globally, must be copied or replicated to all the servers (see Chapter 12 for more information about replication).

Additionally, the application needs to send each SQL query to the partner server "owning" the majority of data required for the query. This is called *collocating* the query with the data required by the statement. Collocating SQL statements with the required data is not a requirement unique to federated servers; it is also required in clustered systems.

Creating Distributed Partitioned Views

Three steps are necessary to define a distributed partitioned view of your database on two or more federated servers:

1. **Create linked servers** Create an image of the database on each of the participating federated servers. Define each member server as a linked server.
2. **Partition your data** Define the member tables that you are horizontally partitioning, including the primary and check constraints.
3. **Creating a distributed view** Define the distributed partitioned views in all the member databases.

Creating Linked Servers

To begin creating a partitioned view, you need to create identically named databases on each federated server. Add a linked server definition on each server for each of its fellow federated servers. You can set up a linked server by running the system-stored procedure sp_addlinkedserver *or* from within Enterprise Manager as follows:

1. Open the Security folder and right-click Linked Servers, then click New Linked Server.
2. Click the General tab, and enter the server name to link to in the Linked Server box. The Server Type must be SQL Server if you want to create partitioned views.

Partitioning Your Data

Once you have decided on the criteria that you need for data portioning (for example, locality or department), a unique CHECK Constraint range on the primary key field in each table on each server must be set to enforce the ranges of data you want each server to own.

The key ranges of the CHECK constraints, on each table in a partitioned view, cannot overlap with the ranges of any other table used. Any given value of the partitioning column must map to only one table on each server. The CHECK constraints can use only the following operators: BETWEEN, AND, OR, <, <=, >, >=, and =.

Creating a Distributed View

The partitioned view is a set of SELECT statements of which the individual result sets are combined into one using the UNION ALL statement. Partitioned views should conform to these guidelines:

- Each individual SELECT statement references one SQL Server base table. The table can be either a local table or a linked table referenced using either a four-part name or the OPENROWSET function. (You cannot use an OPENROWSET function that specifies a pass-through query.)
- No table can be referenced more than once in the view.
- The columns in the same ordinal position in the select list of each SELECT statement in the view definition must be of the same type (including data type, precision, scale, and collation).
- No column can be referenced more than once in the select list of a SELECT statement in the view. Each select list in the view must reference all the primary key columns in the underlying base tables. Each base table has a partitioning column whose key values are enforced by CHECK constraints. The partitioning column must be in the same ordinal location in the select list of each SELECT statement in the view. For example, the partitioning column is always the first column in each select list or the second column in each select list, and so on.
- The partitioning column in each table must *not* allow nulls; it must be a part of the primary key of the table. The partitioning column cannot be a computed column.

- The table must not have timestamp or identity columns. None of the columns can have a DEFAULT constraint. The table cannot have indexes on any computed columns.

Note If the base tables have columns not referenced in the partitioned view, the columns must allow nulls.

Updateable views must also conform to all the following rules:

- INSERT statements referencing the view must supply a value for the partitioning column that satisfies the logic of the CHECK constraint defined on the partitioning column for one of the member tables.
- No UPDATE statement referencing the partitioned view can change the value of the partitioning column.
- There are no restrictions on DELETE statements.

Using and Updating a Distributed View

To create the distributed view depicted in Figure 3.4 , we want to locate the Texas-only warehouse inventory data on SERVER1 and the Florida-only retail store inventory data on SERVER2. Very rarely would there be a need for users at each site to query across both sets of data on a daily basis, but a corporate manager attached to SERVER3 would still be able to generate a cross-location inventory report.

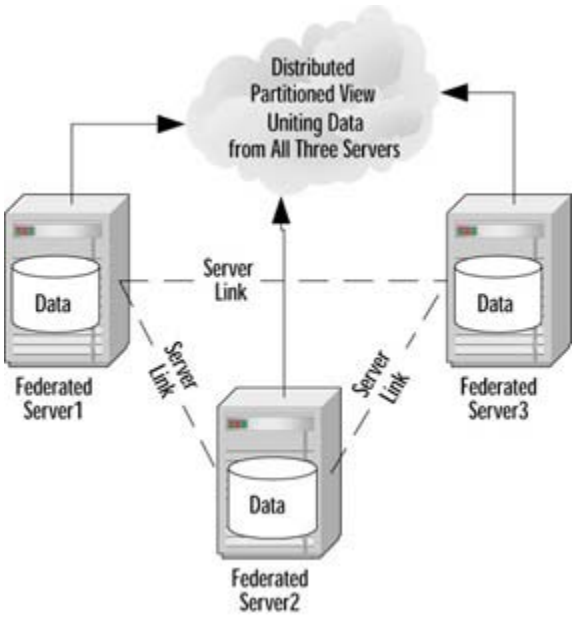


Figure 3.4: Federated servers with partitioned data and links.

Use the following steps to create the partitioned Inventory view:

1. Create a database named InventoryDB on both SERVER1 and SERVER2. When creating tables, set a Check Constraint on the key field STATE that will uniquely partition your data:

```
2.        — On SERVER1:
3.              CREATE TABLE TX_INVENTORY
4.                (STATE  CHAR(2) PRIMARY KEY
5.                                CHECK (STATE='TX'),
6.                … —
7.
8.        column definitions
9.
10.       .)
11.             — On SERVER2:
12.             CREATE TABLE FL_INVENTORY
13.               (STATE  CHAR(2) PRIMARY KEY
14.                                 CHECK (STATE='FL'),
    …—more column definitions.)
```

15. Then add a linked server definition on SERVER1 and SERVER3 for SERVER2:

```
EXEC SP_ADDLINKEDSERVER @SERVER='SERVER2'
```

16. Add a linked server definition on SERVER2 and SERVER3 for SERVER1:

```
EXEC SP_ADDLINKEDSERVER @SERVER='SERVER1'
```

17. Create a distributed partitioned view called VWINVENTORY on SERVER1:

```
18.       CREATE VIEW VWINVENTORY AS
19.                  SELECT * FROM
20.             SERVER1.INVENTORYDB.TABLEOWNER.TX_CUSTOMERS
21.             UNION ALL
     SELECT * FROM SERVER2.INVENTORYDB.TABLEOWNER.FL_CUSTOMERS
```

22. Create a view with the same name on SERVER2 and SERVER3.

Once the view is created, you can update the underlying tables on either or both federated servers by executing T-SQL UPDATE statements directly against the view. For instance, the following code to update the VWINVENTORY view actually updates both the FL_INVENTORY table on SERVER2 and the TX_INVENTORY table on SERVER2 simultaneously.

```
UPDATE VWINVENTORY
  SET LOWSTOCKTHRESHHOLD=50
```

Log Shipping

A new feature of SQL Server 2000 (Enterprise and Developer Editions only) is built-in *log-shipping* support and management utilities.

Tip Log shipping can be implemented manually in SQL Server 7.0. The Microsoft BackOffice 4.5 Resource Kit contains tools and documentation for implementing log shipping on SQL Server 7.0. Look in the online help file SQLTOOLS.CHM for details.

The process of log shipping automatically copies transaction logs from one server to another at specified intervals, effectively keeping the databases on each server synchronized and thereby creating a warm standby server. In the event that the main database server fails, you can instate the backup server can be temporarily in its place by changing its role (if this option is selected during Setup). Log shipping can also be used to support a read-only server for reporting or distributing read-only application functionality.

Log shipping is not as resource-intensive as database replication (see Chapter 12), but it can be used in a similar fashion to maintain a synchronized read-only copy of your main database. You might want to offload some of your company's heavier query processing or reporting to this read-only server or use it for development and/or testing purposes.

Warning Be aware that switching to a log shipping standby server is a *manual* process. The fail-over is not automatic; you must use the Log Shipping Monitor to perform the role switch-over. In addition, you will need to manually change the server name that applications use to connect to the backup server.

The three major roles in a log-shipping system are the source server, the destination server, and the monitor server (see Figure 3.5). The *source server* contains the active read/write version of the database. The *destination server* asynchronously restores the source server transaction logs, which have been backed up to a network location. The *monitor server* is used to monitor log-shipping information and events. Conceivably, all three elements could be installed on the same server, but performance would likely be impacted. Furthermore, you would lose the main benefit of log shipping due to the server becoming a single point of failure.

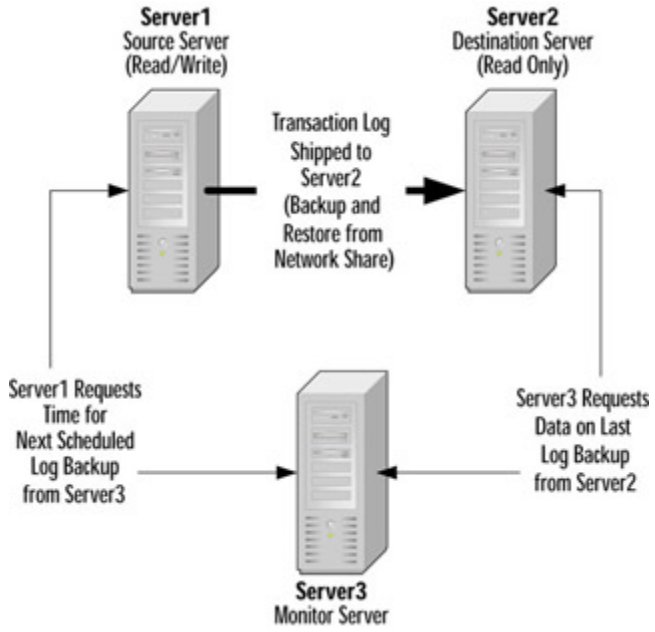


Figure 3.5: The three server roles in log shipping: source, destination, and monitor.

Setting Up Log Shipping

Your SQL Server login account must be a member of the sysadmin server role to set up log shipping. The account used to start the MSSQLServer and SQLServerAgent services must also have permission to modify the log-shipping jobs on both the source and destination servers.

Note You can configure a server running SQL Server 7 Enterprise (SP2 only) as your log-shipping destination. Execute the following stored procedure on the SQL 7 Server to enable log shipping between the two versions:

```
EXEC sp_dboption 'database name', 'pending upgrade', 'true'
```

This stored procedure allows SQL 7 to provide limited support for new functionality found in SQL Server 2000, to be used temporarily during upgrading. However, when the "pending upgrade option" is set to TRUE, indexes and statistics cannot be created in the database. Consult the SQL 7 SP2 documentation for more information.

In Enterprise Manager, connect to the server you want to use as your Source Server:

1. Under Tools, launch the Database Maintenance Plan Wizard. Select Database Maintenance Planner *or* select New Maintenance Plan under the Management folder, Database Maintenance Plans *or* select an existing plan under the Management folder, Database Maintenance Plans, and click its Properties tab.

2. From the Select Databases screen (see Figure 3.6), select These Databases, and then check the box next to the database you want to log ship. (Only one database can be selected at a time, and you can't select the same database twice.)

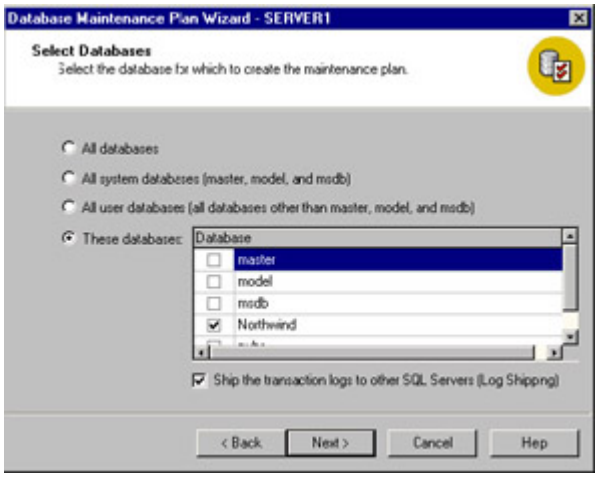


Figure 3.6: The Select Databases dialog box.

3. Check Ship Transaction Logs to Other Servers (Log Shipping), then click Next.
4. Specify any additional database maintenance plan options or continue clicking Next until you reach the Specify the Log Shipping Destinations screen.
5. Click Add, and select a destination server name and database name on the Add or Edit Destination Database screen (see Figure 3.7), or you can select Create New Database if one doesn't already exist. Click Next.

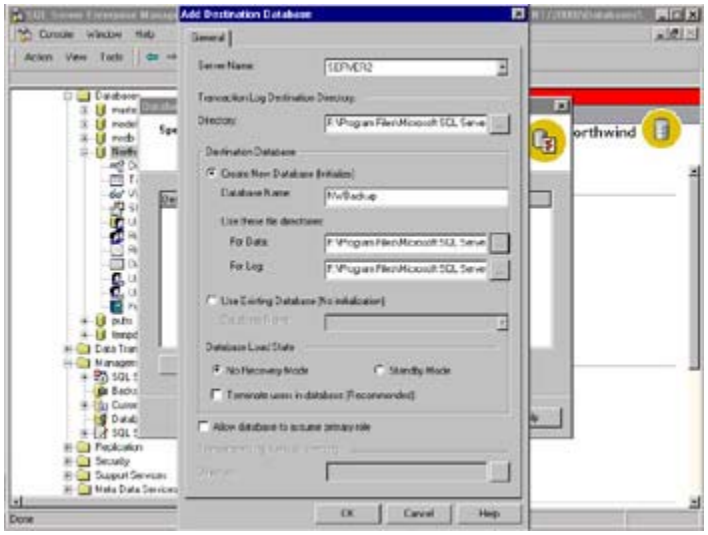


Figure 3.7: Adding or Editing the log-shipping destination database.

6. Check the Allow Database to Assume Primary Role box, and specify a transaction log backup directory if you would like the destination server to be capable of assuming the role of source in the event of source failure.

7. From the Initialize the Destination Databases screen, choose Perform a Full Database Backup Now or Use Most Recent Backup File. If you choose to use a backup file, select it from the browse list. Click Next. Note that you can store transaction logs only to hard disks during log shipping. The backup-to-tape option is not available.
8. From the Log Shipping Schedules screen (see Figure 3.8), view the default schedule, and click Change if you need to make modifications to the schedule. The Copy/Load Frequency box allows you to set the frequency in minutes with which you want the logs to be backed up and restored from source to destination. The Load Delay box sets the length of time to wait between backing up and restoring a transaction log.

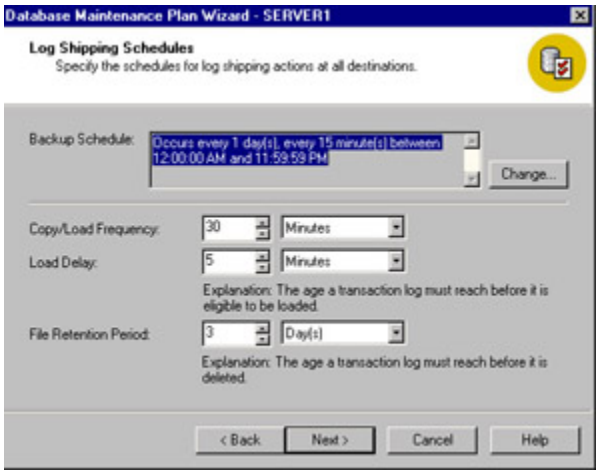


Figure 3.8: Setting the log-shipping schedules.

9. Set the File Retention Period to specify how long to keep logs in the backup directory before deleting them. Click Next.
10. From the Log Shipping Thresholds screen (see Figure 3.9), set the backup alert threshold (the maximum time to wait between backups before sending an alert message to the monitor server) and the out-of-sync alert threshold (the maximum time to wait between backups and restores before sending an alert). Click Next.

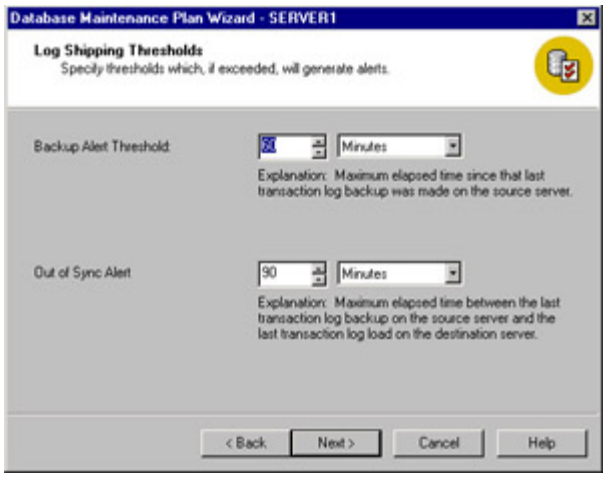


Figure 3.9: Specifying the log-shipping alert thresholds.

11. On the Specify the Log Shipping Monitor Information screen (see Figure 3.10), select the server that will perform the role of monitor server, along with its SQL Server or Microsoft security information. Click Next.

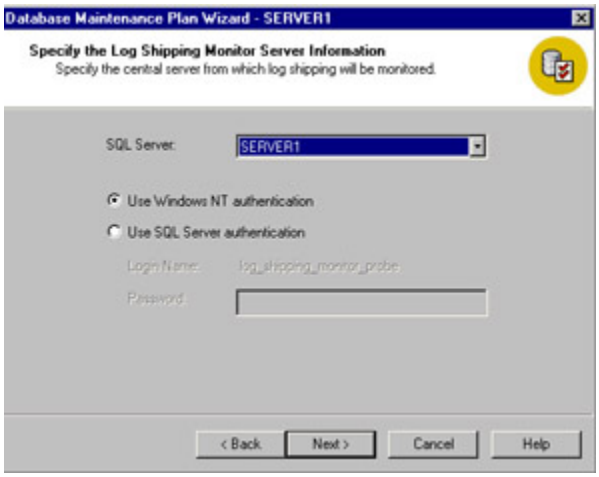


Figure 3.10: Specifying the log-shipping monitor server information.

12. If this is a new database maintenance plan, enter a name for it, and click Finish to save it and exit the wizard.

Stopping Log Shipping

To stop log shipping, select an existing plan under the Management folder, Database Maintenance Plans, and click its Properties. On the Log Shipping tab, click Remove Log Shipping.

Warning This action completely removes your log-shipping configuration from the primary, secondary, and monitor servers.

Monitoring Log Shipping

The monitor server can be used to view detailed log-shipping information for the source server and destination server(s). Monitored data includes when the last backup and restore took place and backup/load timeout or failure notices. Alert information can also be filtered or suppressed. If a destination server was previously configured to allow it to change roles, you can also use the Log Shipping Monitor to change it to a source server in the event of system failure.

To view the status of log shipping:

1. Connect to the Monitor Server, expand its Management folder, and select Log Shipping Monitor.
2. Click the log shipping server pair that you want to monitor, and select Properties.

From the Log Shipping Pair Properties screen Source or Destination tabs (see Figures 3.11 and 3.12 , respectively), you can view the backup schedule, view the last backup and restore times, and view alerts and/or suppress them as necessary.

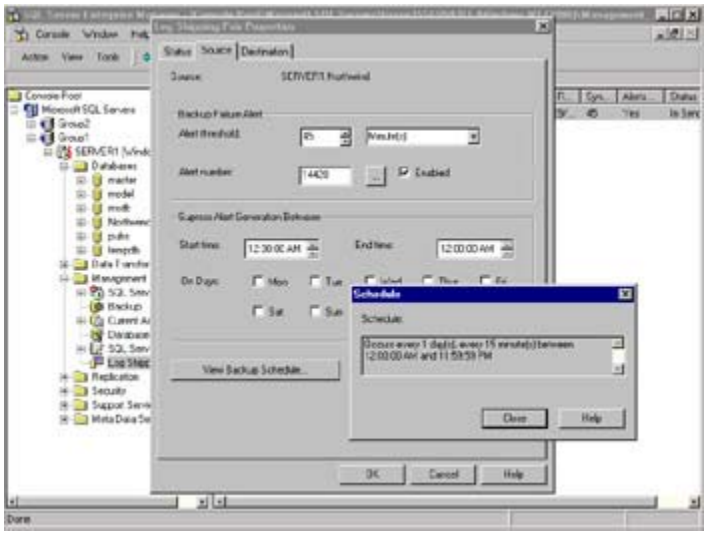


Figure 3.11: Modifying the log-shipping source schedule.

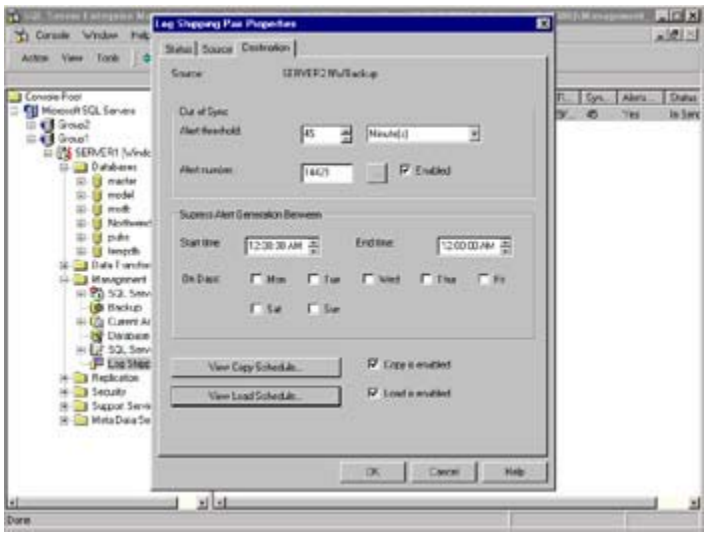


Figure 3.12: Modifying the log-shipping destination schedule and other properties.

Indexed Views

An especially powerful feature in SQL Server 2000 is the ability to place an index on a view. Adding indexes to views significantly enhances performance, in the same manner that adding an index to a table does. In previous versions of SQL Server, many developers refrained from using views to query large data sets, because the sequential access gave poor performance results.

View indexes are stored in the database in the same manner as table indexes and can be either clustered or nonclustered. A clustered index has specific, ordered meaning that can be grouped. A nonclustered index is not necessarily ordered or sorted according to any specific guidelines. Like a clustered index on a table, the complete result set of the view is stored in the database, but the clustered index's b-tree structure contains only the key columns. When the underlying tables of a view are modified, its indexes are modified as well. This modification adds some overhead to the system, but the performance gain for view querying is well worth it. *Indexed views are a feature only available in SQL Server 2000 Enterprise Edition.*

SQL Server uses indexed views in two different ways:

- An indexed view can be called directly from a query, just like any nonindexed view, but instead of running its underlying SELECT statement dynamically, it uses the index to display the view results so that queries appear to be almost instantaneous.
- SQL Server 2000 Query Optimizer can automatically evaluate queries to see whether any existing indexed views can fulfill them. If it finds an indexed view that will work, it uses it, without the user having to specify it.

Situations that benefit most from indexed views include:

- Data marts, data warehouses, decision support, data mining, OLAP applications
- Views in which data is relatively static
- Views that join two or more large tables
- Views that aggregate data for a large number of rows
- Repeated patterns of queries
- Any combination of the preceding situations

Indexed views should be not be used in cases in which the increased overhead of modifying the indexes is too intense (for example, in intense OLTP applications that have a very percentage of INSERTS, UPDATES, and DELETES to the underlying tables).

Requirements for an Indexed View

In order for a view to be indexed, these requirements must be met when the view is created:

- Prior to the CREATE VIEW statement or the CREATE TABLE statements for the underlying tables, you must set the following session parameters:

```
SET ANSI_NULLS ON
SET QUOTED_IDENTIFIER ON
```

- The WITH SCHEMABINDING option must be used in the CREATE VIEW statement and for creating any user-defined functions used by it. This option binds the view to the same schema as the underlying tables. In addition, the view must be located in the same database as its underlying tables and the owner/schema of both view and tables must be the same. As such, only one-part names are allowed in creating the views; no databases or schema names can be specified as prefixes.
- The view query can reference only base tables, not any other views.

The following T-SQL syntax must be adhered to in your CREATE VIEW statement:

- You *cannot* use the wildcard SELECT * statement; columns must be explicitly listed.
- You *cannot* use SELECT DISTINCT, SELECT TOP, or SELECT COUNT(*).
- You *cannot* use ORDER BY, HAVING, CUBE, or ROLLUP.
- Derived tables, subqueries, UNIONs, OUTER JOINs, and self-joins are not allowed.
- Complex aggregate functions MAX, MIN, AVG, and the like are not allowed. Simple functions such as SUM(x) and COUNT_BIG(x) are allowed; however, no functions can be specified unless GROUP BY or COMPUTE are included.
- Rowset functions are not allowed.
- The *float* data type cannot be used for an index key.
- The view cannot contain text, ntext, or image columns.

The CREATE INDEX statement on a view requires that:

- The user creating the index must be the view owner.
- All the following session parameters must be set:

```
SET ANSI_NULLS ON
SET ANSI_PADDING ON
SET ANSI_WARNINGS ON
SET ARITHABORT ON
SET CONCAT_NULL_YIELDS_NULL ON
SET QUOTED_IDENTIFIERS ON
SET NUMERIC_ROUNDABORT OFF
```

- If a GROUP BY clause exists, the clustered index key must be a field in the GROUP BY list.

Tip In order to select the ideal unique clustered index (or any of the nonclustered indexes) for an indexed view, the index should be selected at the same time indexes are selected for the underlying table(s) used in the view. It is important not to add

indexes that are redundant to the underlying table(s) or to the view itself. Try to keep the index column size as conservative as possible in order to reduce the overall size of the index. The Index Tuning Wizard is an especially good tool for evaluating and selecting the view and underlying table indexes.

Creating an Indexed View

The first index created on a view must be a unique clustered index. Additional nonclustered indexes can be created afterward. You need to make sure that all the requirements in the previous section have been met before creating the underlying tables, the view itself, and the clustered index.

The following example creates an indexed view on the Northwind database to quickly pull up a list of products currently in stock by product name:

```
USE NORTHWIND
SET ANSI_NULLS ON
SET QUOTED_IDENTIFIER ON
GO

CREATE VIEW VWPRODUCTSINSTOCK
WITH SCHEMABINDING
AS
SELECT PRODUCTID, PRODUCTNAME, UNITSINSTOCK
FROM DBO.PRODUCTS
WHERE UNITSINSTOCK>0
GO

CREATE UNIQUE CLUSTERED INDEX VWPRODUCTSINSTOCK_IDX
ON VWPRODUCTSINSTOCK (PRODUCTNAME)
GO

SET ANSI_NULLS OFF
SET QUOTED_IDENTIFIER OFF
GO
```

Summary

SQL Server 2000 Enterprise Edition combines powerful new scalability and high-availability features to produce a platform capable of supporting larger and more complex applications for the 24 x 7 application environment of today. No longer should you think simply in terms of scaling up to improve application performance. Instead, you can consider taking advantage of SQL Server's ability to scale out.

Distributed partitioned views give you the power to scale out by storing portions of your database on multiple, federated servers to achieve optimal query performance. These compartmentalized chunks of data are much faster to access on the local level, while they continue to allow global accessibility. Each federated server should maintain at least 80 percent of the data to be accessed by its "local" applications. Partitioned views are also updateable, allowing data to be updated in the underlying tables across multiple servers simultaneously. The ability to create indexes on views is a welcome addition to SQL Server 2000. Indexed views enhance online application productivity by speeding up query time in large databases. View indexes are stored in the database in the same way; table indexes and results are not generated "on the fly," like traditional views. View indexes can be utilized by directly querying against indexed data, or they can be selected automatically by the SQL Server Query Optimizer.

SQL Server 2000 has proven itself a major contender in the realm of fault tolerance because its fail-over clustering has become even more reliable and manageable. Fail-over clustering runs on top of the Microsoft Cluster Service platform, which requires identical hardware and software to be run on each member server. The SQL Server Setup Wizard recognizes the existence of MSCS and prompts you to configure SQL Server Fail-over Clustering.

The new log-shipping feature can be used by administrators who require a less expensive alternative solution to SQL Server Fail-over Clustering. Log shipping allows transaction logs from a main read/write source server to be backed up to a network share and restored on an automated schedule to a read-only destination server. In the event of a failure in the source server, the administrator has the option of converting the destination server to read/write service. Unlike MSCS, this fail-over is a manual process, since the log-shipping servers do not have the same names and application connection strings would need to be redirected to the new server's name.

SQL Server scalability has improved by leaps and bounds. Since its release into beta testing, SQL Server 2000 has produced consistently high TPC-C and TPC-H benchmark statistics. Recent benchmarks give SQL Server the decided edge over its competition for both transaction processing and decision support.

FAQs

Q. After upgrading my SQL Server 7.0 Fail-over Cluster, I cannot open a database connection to either server.

A. Check the instance names. If no instance name was assigned to either server during the upgrade, they are both called "default" and the database engine cannot access either of them. Rename one or both of the instances on the server by rerunning SQL Server Setup. In addition, be sure that the instance name on the virtual server is not the same as the instance name on any individual node in the cluster.

Q. I have set up a distributed partitioned view called VWINVENTORY, which contains inventory data from three servers, each having a warehouse using the column WHID as my check constraint. I want to insert a new product record, which will be carried by two of the three warehouses.

A. You can do this either of two ways. Either insert the new product record directly into the underlying tables on each server, as follows:

```
INSERT INTO Server1.InventoryDB.dbo.WH1_Data (WHID,

COLUMN LIST…

)
VALUES ('WH1', —ADDITIONAL COLUMN VALUES.. )

INSERT INTO Server3.InventoryDB.dbo.WH3_Data (WHID, COLUMN LIST…)
VALUES ('WH3', —ADDITIONAL COLUMN VALUES…)
```

or insert the same records into the view:

```
INSERT INTO VWINVENTORY
VALUES ('WH1', —ADDITIONAL COLUMN VALUES )

INSERT INTO VWINVENTORY (WHID, COLUMN LIST…)
VALUES ('WH3', —ADDITIONAL

COLUMN VALUES

)
```

Q. I have a large database running on a single server with four processors and 2GB RAM. The motherboard won't support additional processors or RAM, so I'm considering scaling out. What SQL Server features can I utilize to provide improved application performance and fault tolerance?

A. You can use distributed partitioned views to load-balance queries across several servers.

Q. I am trying to configure log shipping, but when I try to select a destination server name, the server select list is blank.

A. The destination server must be registered in Enterprise Manager in order to appear in the list. It must also be running either SQL Server 2000 Enterprise Edition or SQL Server 7.0 Enterprise Edition with SP2 installed. In addition, your SQL Server login account must be a member of the sysadmin server role on all involved servers.

Chapter 4: Designing and Creating SQL Server Databases

Introduction

Every organization has data close to its heart. Storing and caring for data are the roles of SQL Server. SQL Server's robust RDBMS architecture supports hosting multiple, distinct databases and, since version 7.0, provides native file system support, simplifying the management of database files. Planning for the location and growth of these files is an important process in designing and creating your database.

This chapter discusses the architecture of SQL Server, various storage systems for SQL database files, and how to design an efficient database file structure for your application.

Creating databases in SQL Server can be as simple as responding to dialog box prompts using the Create Database Wizard or as configurable as the T-SQL CREATE DATABASE statement. We review the options available for creating databases and configuring options such as database support for various collations, a new option in SQL Server 2000. Several features are available for tasks such as moving databases and supporting database growth with file autogrow and multiple data files.

Before you can begin creating the physical database and storage files, you need to understand the database structure and its requirements. This chapter reviews the architecture of SQL Server as well as the modeling techniques and tools available to design and implement your database solution. Having a clear understanding of the database model is essential to planning and modifying your physical database.

SQL Server 2000 Architecture

SQL Server 2000 consists of numerous components that interact to provide complete database application capabilities, including relational database management, OLAP, data mining, full-text indexing, data import and export, and replication, as well as client access, as depicted in Figure 4.1 . In the later chapters of this book, we review each of these components in detail and assist you in configuring and using them in your applications. This chapter begins by exploring the base components of SQL Server and its databases.

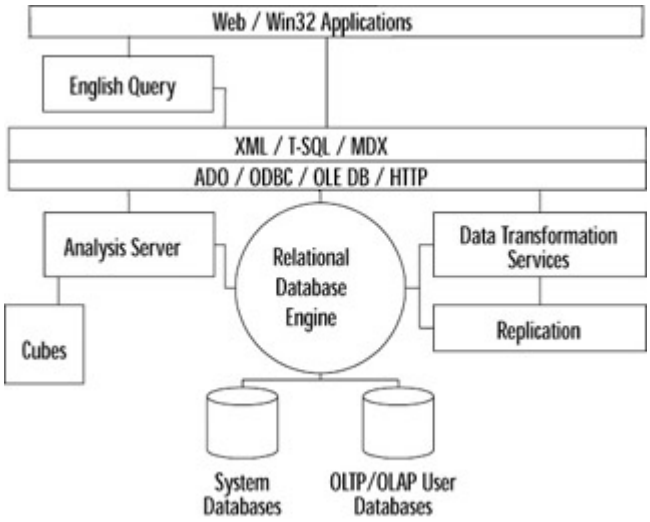


Figure 4.1: SQL Server architecture overview.

SQL Server 2000's relational database architecture is highly scalable and reliable and continues to meet the growing demands of thousands of customers with databases into the terabytes and users in the thousands. The foundation of every SQL Server solution begins with the same component: the database. Each SQL Server instance can support up to 32,767 databases, each with a maximum database size of 1,048,516 terabytes (TB). SQL Server 2000 has also increased the maximum size of its log files from 4TB in version 7.0 to 32TB in SQL Server 2000, offering greater transactional capacity. Table 4.1 provides the maximum database properties of SQL Server 2000.

Table 4.1: SQL Server Maximum Capacities

Object/Property	**SQL Server 2000**
Instances per server	16
Databases per instance	32,767
Filegroups per database	256
Files per database	32,767
Database size	1,048,516TB
Data file size	32TB

Table 4.1: SQL Server Maximum Capacities	
Object/Property	**SQL Server 2000**
Log file size	32TB
Total database objects	2,147,483,647
Columns per table	1,024

Relational Databases

Although there are numerous methods of storing information, the relational database model has grown to be recognized as the most efficient data storage model. Relational databases are based on the need to efficiently organize and store data. Data in a relational database are organized as entities and stored in tables. Each table consists of attributes, which result in the columns that make up a table.

The process of identifying the primary data items or entities and efficiently laying out these tables and their attributes is accomplished through a process called *normalization* . The process of normalization works to reduce data redundancy and derivation, producing efficiency in both storage and data management. By eliminating redundant and derived data (data that are the result of other attributes), the physical storage requirements are reduced. Additionally, the task of managing multiple copies of identical data as well as computed or derived data is eliminated, resulting in a more accurate and manageable database. Complex data items can be broken down into multiple tables to produce a normalized database model, as shown in the example in Figure 4.2 . This simple model depicts a portion of an order-entry database. This collection of "related" tables makes up a relational database.

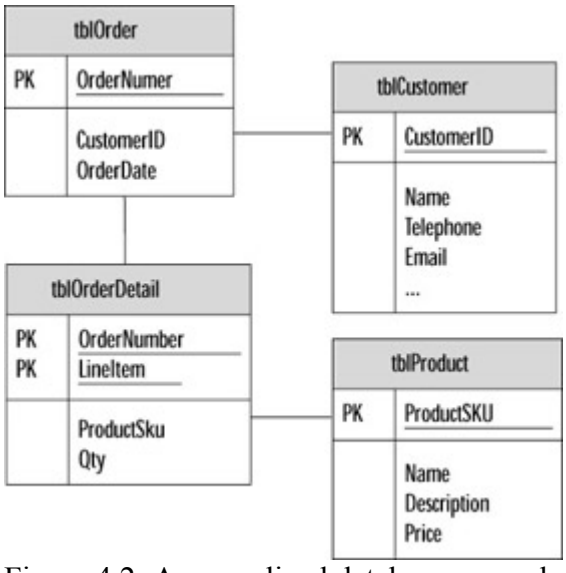


Figure 4.2: A normalized database example.

SQL Server supports the requirements of a RDBMS by providing the logical and physical storage architecture that are needed. At the core of the logical storage architecture of SQL Server 2000 are databases, tables, and columns. The database represents the overall data grouping and is the foundation of SQL Server applications. Tables store individual data entities and consist of columns (attributes) that store the actual data item values. Beyond the base components are several higher-level database items, commonly referred to as *database objects* , such as views, triggers, indexes, keys, defaults, constraints, stored procedures, user-defined data types, and user-defined functions. Each of these objects plays a particular role in controlling the integrity of the data or effectively delivering it for application use.

The following list outlines each of the components of a relational database solution in SQL Server 2000:

- **Database** The database is the primary object and contains all the remaining database objects, such as tables, views, and the like.
- **Table** Each table represents a data item or entity. Tables are typically logical data containers, such as a customer table or an order table in an order-entry database solution.
- **Column** A column is a property of a table and represents an attribute or data item about the table.
- **View** A view is often called a *virtual table* and is typically used to combine several tables in to a meaningful representation of the data. Views do not physically store the data; rather, they provide a combined presentation of the underlying tables.
- **Trigger** Triggers are routines or stored procedures that execute automatically when an INSERT, UPDATE, or DELETE action occurs against a table. Triggers are often used to enforce business rules in a database application. For example, a trigger could send an e-mail message when the Orders table receives new data.
- **Index** An index relates to a table and speeds the retrieval of data from the table by providing a representation of the data that is more efficient for locating the requested information.
- **Key** A key is a property of a table and is one or more columns that uniquely identify each record or row in the table.
- **Default** A default specifies the value of a column during an insert if an explicit value is not supplied.
- **Constraint** Constraints specify the valid values for a specific column and are commonly used to enforce integrity rules.
- **Stored procedure** A stored procedure is a predefined group of Transact-SQL statements that is commonly used as a routine to manipulate or perform complex filter operations in order to retrieve specific data.
- **User-defined data type** A user-defined data type, or UDT, is used to enforce exact data types across multiple tables and is based on a standard data type—for example, a Social Security number (SSN) data type that represents an 11-character (nnn-nn-nnnn) data type.

- **User-defined function** A user-defined function is a reusable logic set that can be called to perform set actions and return a given result. User-defined functions are similar to stored procedures except that they can return data type results, allowing them to be used in line with T-SQL statements.

SQL Server System Databases

Every installation of SQL Server 2000 includes six databases that are created during the setup process. Four of these databases are used by SQL Server to manage server and database configuration information as well as service user requests, such as query activity. These system databases are the Master, Model, msdb, and TempDB databases. Each of these system databases is required by SQL Server 2000 and cannot be renamed or removed without disabling SQL Server. The remaining two databases, Pubs and Northwind, are sample databases that contain fictitious data. Many of the examples that Microsoft and other organizations provide use these sample databases because of their availability with every SQL Server installation. The sample databases can be removed from SQL Server 2000 without affecting SQL Server's operations.

Master

The Master database is the primary configuration database in SQL Server. It contains information on all the databases that exist on the server, including the physical database files and their locations. The Master database also contains SQL Server's configuration settings and login account information. The following list outlines the information contained in the master database:

- Server Registrations and Remote Logins
- Local Databases and Database Files
- Login Accounts
- Processes and Locks
- Server Configuration Settings

Given its importance to SQL Server, a current backup of the Master database is critical to any server recovery. The installation process creates the Master database (master.mdf) and master database log file (mastlog.ldf) in the Program Files\Microsoft SQL Server\Mssql\Data directory. In the event that your Master database is corrupted or destroyed and cannot be restored from backups, you can rebuild the Master database to its default state. Microsoft provides a utility named rebuildm.exe that is located in the Program Files\Microsoft SQL Server\ 80\Tools\Binn subdirectory. The tool provides a GUI that allows you to specify the source of the existing database file and the destination server, as shown in Figure 4.3 . Using this utility, you can use the Master database from the SQL Server installation CD-ROM to replace the missing or corrupted Master database on your system. After you have rebuilt the Master database you will have to manually make any configuration changes and server and database registrations and recreate logins to restore your server to its previous state.

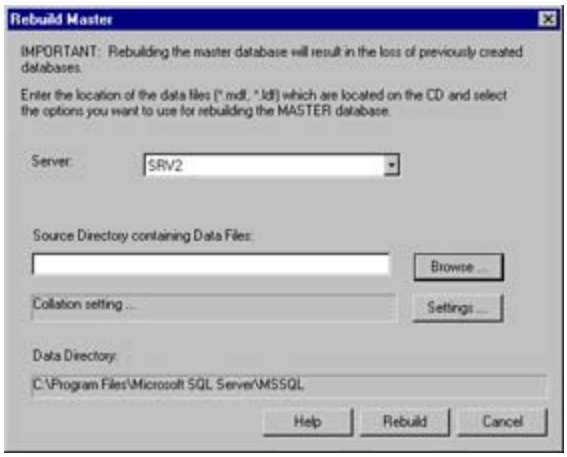


Figure 4.3: The Rebuild Master Database utility.

TempDB

SQL Server uses the TempDB database for working storage of temporary tables and temporary stored procedures. Temporary stored procedures and tables can be created by users, applications, or the system to support query requests. Notice the use of the word *temporary* here. The TempDB database is recreated each time SQL Server starts, so the TempDB should *not* be used for persistent data storage. Temporary tables and procedures that are generated by SQL Server are automatically dropped when connections are closed; therefore, under no connection activity, the TempDB is empty. All databases and processes in SQL Server share TempDB for working storage. The TempDB database file (tempdb.mdf) and log files (templog.ldf) are located in the Program Files\ Microsoft SQL Server\Mssql\Data directory.

Note TempDB is initially set to 8.0MB with autogrow enabled so that space is acquired as needed. Because the TempDB is recreated each time SQL Server starts, if your application continually requires the TempDB to autogrow to meet its needs, you can modify the TempDB's initial database size to eliminate this processing overhead. You can use the Database Properties dialog box or the ALTER DATABASE command to accomplish this task.

msdb

The msdb database is a system database that is used by several SQL Server components such as the SQL Server Agent service. In addition to SQL Server Agent configuration and task information, replication, log shipping, and maintenance plan data are stored in the msdb database. The following list outlines the information contained in the msdb database:

- SQL Server Agent information such as jobs, job history, job schedules, operators, alerts, and notifications
- Database maintenance plan information such as maintenance plan jobs and history
- Replication publishers and distributor information

- Log shipping configuration and monitoring information

The msdb database file (msdb.mdf) and msdb log files (msdb.ldf) are located in the Program Files\Microsoft SQL Server\Mssql\Data directory. Due to the amount of configuration information stored in the msdb database, the database should be routinely backed up. If the msdb database becomes damaged or corrupted, you can rebuild a default msdb using a SQL build script installed during the installation process. The msdb install script filename is instmsdb.sql and it is located in the Program Files\Microsoft SQL server\Mssql\Install directory. If the msdb database does not exist, the script will create it first and then build all the necessary database objects. You can execute the instmsdb.sql script using the Query Analyzer tool. As the script processes, it will display status messages and report any errors that occur. After you have rebuilt the msdb database, you will have to reconfigure any scheduled jobs, maintenance plans, log shipping, and replication configurations.

Model

The Model database is the template database that SQL Server uses to create new databases. Each time you create a new database in SQL Server, the contents of the Model database are copied to the new database to establish its default objects, including tables, stored procedures, and other database objects. The Model database is required even if you do not intend to create any new user databases. Each time SQL Server starts, the TempDB is recreated using the Model database as its template. By default, the Model database is empty when it is created. The Model database file (model.mdf) and Model database log files (modellog.ldf) are created in the Program Files\Microsoft SQL Server\Mssql\Data directory during the installation process.

Pubs

The Pubs database is one of the two sample databases that are included with the SQL Server installation. It is modeled after a book publishing company and demonstrates some of the options available in SQL Server. Many of the examples that Microsoft and others provide use this database due to its availability. The Pubs database can be removed without affecting SQL Server's operations. You can examine the Pubs database to view sample tables, stored procedures, and user-defined data types. The Pubs database offers sample data and provides an excellent "playground" for new SQL developers to become comfortable with SQL Server concepts, without having to create their own sample database. The Pubs database file (pubs.mdf) and pubs log file (pubs_log.ldf) are located in the Program Files\Microsoft SQL Server\Mssql\Data directory.

If you modify the Pubs database and decide to return it to its original state, you can use a provided SQL build script, instpub.sql. The instpub.sql script is located in the Program Files\Microsoft SQL server\Mssql\Install directory. The SQL script will drop the existing Pubs database, so verify that no user connections are open to it or you will receive an error message from the script. You can execute the instpub.sql script using the Query

Analyzer tool. As the script processes, it will display status messages and report any errors that occur.

Northwind

The Northwind database, the second of the two sample databases that are included with SQL Server, is modeled after a fictitious company that provides sales data for importing and exporting specialty foods. This popular sample database is commonly used by Microsoft and others. This book also uses the Northwind database throughout its examples. As you can with the Pubs data-base, you can browse Northwind's table, view, and stored procedures and modify it to examine additional functionality in SQL Server. The Northwind database file (northwind.mdf) and Northwind log file (northwind.ldf) are located in the Program Files\Microsoft SQL Server\Mssql\Data directory.

If you modify the Northwind database and need to return it to its original state, you can use a provided SQL build script, instnwnd.sql. The instnwnd.sql script is located in the Program Files\Microsoft SQL server\Mssql\Install directory. The SQL script will drop the existing Northwind database, so verify that no user connections are open to it or you will receive an error message from the script. You can execute the instnwnd.sql script using the Query Analyzer tool. As the script processes, it will display status messages and report any errors that occur.

Physical Storage Architecture

Beginning with SQL Server 7.0, Microsoft made a significant shift in physical storage architecture by moving to native file system support for data and log files. Databases in SQL Server 2000 are stored in filegroups that contain individual data and log files. Each database is made up of primary, and optionally, secondary data files as well as log files. These files can easily be identified by their default file extensions: primary data file (*.mdf), secondary data file (*.ndf), and log file (*.ldf). Data and log files can be stored on any uncompressed FAT or NTFS drive.

With the exception of log files, data in SQL Server files are stored in units called *pages* . Each page stores 8K of data, allowing for a maximum of 8060 bytes on each page (96 bytes is used for each page header). Due to the 8060 byte page data limit, the maximum record size is also 8060 bytes, which includes all data types except *ntext, text,* and *image* data. *Text, ntext,* and *image* data are stored in a collection of separate pages that allow up to 2GB of information per data item. A pointer to these data pages is stored with the source record.

These pages are combined into groups of eight and allocated to tables and indexes in SQL Server. Each group of eight data pages is called an *extent* . The next obvious question here is, what if a table doesn't require the eight 8K pages, or 64K, of storage? SQL Server addresses this issue by supporting two different classes of extents: *uniform extents,* which are entirely allocated to a particular object, and *mixed extents,* which allow

a single eight-page extent to be shared by up to eight objects total. This relationship among data files, pages, and extents is depicted in Figure 4.4 .

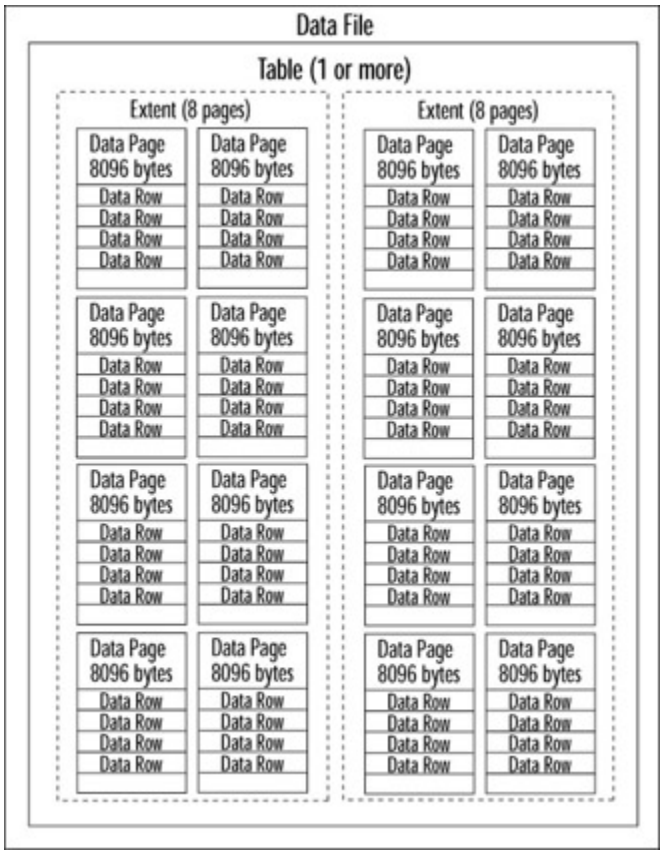


Figure 4.4: Data storage architecture.

Filegroups

Every database that is created contains a PRIMARY filegroup that, by default, contains all data files. *Filegroups* are a means of categorizing or grouping data files into units that can be distributed across multiple physical disks and administered individually. This feature allows specific tables and indexes as well as *text, ntext,* and *image* data to be assigned to a specific filegroup.

When you assign a database object or data type such as *text* to a filegroup, the data are stored in available space in the data files that are assigned to the filegroup. By dispersing database storage across multiple drives through the use of filegroups and data files, your database can enjoy greater throughput.

For example, you might want to locate two highly used tables on different physical disks to increase response time. This is a common practice in laying out your physical database structure and requires intimate knowledge of your data-base and the applications that will use it. To accomplish this task, you create two filegroups and then two data files, each located on a separate physical disk. Assign each data file to a different filegroup. Lastly,

create your new tables or edit existing tables and assign them to each filegroup. Later in this chapter we step through this task in detail.

As we reviewed earlier, each database in SQL Server 2000 is limited to 256 filegroups. Each filegroup, however, can contain up to 32,767 data files, so the ability to segment your database across multiple disks is tremendous. In large, high-activity database solutions, this type of segmentation is essential to performance.

Data Files

There are two types of data files in SQL Server: primary and secondary. When you create a new database, a single *primary data file* is created to store all your data. This primary data file is, logically, located in the PRIMARY filegroup, which is also automatically created and set as the default filegroup for the database. New objects that are created in the database are allocated to the PRIMARY filegroup by default, unless you specify otherwise. Data files can belong to only one filegroup.

In comparison to filegroups that represent logical groups of database objects, data files represent the actual physical storage files. Primary data files are stored by default with the .mdf extension; *secondary data files* are stored with the .ndf extension. These extensions are not required but their use is recommended in order to recognize the physical files on the disk. The actual physical file locations are stored as properties of the database in system tables located in both the Master database and the local database. By default, data files are configured to autogrow at 10 percent. Autogrow, a feature new with version 7.0, allows the data file to automatically size itself to store additional information as required. If we specify 10 percent autogrow, a 10MB data file would allocate additional data in 1MB increments. The ability also exists to restrict the autogrow capabilities to a maximum megabyte size. This is helpful if you are hosting multiple database solutions on a single server and you need to partition the available space among databases.

Transaction Logs

Every database contains at least one data file and one log file. The log file contains the *transaction log,* which records information that describes the changes that are applied to data in the database. The log records contain the beginning of each transaction, the commit or rollback state of the transaction, the data that were changed, and the information necessary to undo the transaction, such as previous column values. Additional events such as page allocations and index events are also recorded in the transaction log. This “record of changes” allows you to return the database to a specific point in time and guarantees data integrity by playing forward each change in its original sequence.

Transaction logs do not contain pages like data files; rather, they store information as log records. By default, log files are physically stored with the *.ldf extension. Similarly to data files, the .ldf extension is not required, but it is recommended for easy identification

of log files. Log files are configured by default to autogrow with 10 percent file growth. As with data files, these measures can be adjusted, restricting the maximum file growth to a megabyte limit.

Indexes

As you'll recall from the previous section , "Relational Databases," indexes are a database component that allow for the fast retrieval of data from tables. In detail, indexes also incur additional storage requirements. Understanding how indexes work is important to creating and allocating them to data files.

SQL Server 2000 allows for indexes to be created on tables and views. Creating indexes involves selecting from one to 16 columns that can be used by T-SQL logic to decrease the amount of searching necessary to locate existing data in a table. Without indexes, a process called a *table scan* occurs, which means that the query processor has to go through each record in the table individually to find records that match the selection criteria. This process is similar to locating specific information in this book. You can use the table of contents or the index to find specific topics easily, but without them you would have to flip through every page until you found what you were looking for. With this analogy in mind, you can see the immediate value of indexes. In this chapter, we don't cover the design process of selecting columns for indexes to maximize performance, but understanding them is necessary to define our storage requirements, filegroups, and data files.

There are two types of indexes in SQL Server; each has uniquely different storage requirements and effects on table data. These two types are *clustered* and *nonclustered indexes* :

- **Clustered indexes** Each table can have one clustered index. The reason for this limitation is that clustered indexes physically resort the data, and the table data are located in the pages that hold the clustered index. With this in mind, consider an example in which you have created a table and allocated it to filegroup FG1 and the data are located on Disk 1. You then create a clustered index on the table and allocate it to filegroup FG2 and the data are located on Disk 2. If you query the table, which disk is actually accessed? The answer is Disk 2 because the actual data are located in the clustered index that has been allocated to Disk 2. Understanding that clustered indexes physically sort and contain the actual table data is important to positioning them with filegroups for overall performance. The first index created on a view is a clustered index and has the same effects as a clustered table index.
- **Nonclustered indexes** Each table or view can have up to 249 nonclustered indexes. The opposite of clustered indexes, nonclustered indexes do not physically sort the underlying data; rather, they provide a pointer to data at the matching index point. If the underlying table does not have a clustered index (called a *heap*), the pointer is made up of the file identifier, page number, and row number of the data on the page. If the underlying table does contain a

clustered index, the pointer is the *key* (columns selected for the index) from the clustered index.

SQL Server Services

As with many applications designed to run on Windows NT and 2000, SQL Server incorporates five application services to manage its databases and complete its tasks. These services are similar to other services in Windows NT and 2000 and operate independently of any local users, allowing SQL Server to operate without user interaction. The five services are SQL Server service, SQL Server Agent service, MSSQLServerADHelper service, Microsoft Distributed Transaction Coordinator (MS DTC) service, and Microsoft Search Service. Each of these services is a component of the application responsible for receiving instructions, performing its tasks, and returning desired results as necessary. If you have installed the Analysis Services component of SQL Server, an additional service, MSSQLServerOLAPService, is installed.

You can control the status of each of these services through the SQL Server Service Manager or the Services applet in Windows NT or 2000. The SQL Server Service Manager is available from the Microsoft SQL Server program group or by double-clicking the Service Manager icon located in the system tray, as shown in Figure 4.5 . To manage each of these services from the Services applet in Windows 2000, select the Services icon from the Start | Programs | Administrative Tools group. The SQL Server and SQL Server Agent service names in the services list are slightly different from the names used by Service Manager. The SQL Server service is named MSSQLSERVER and the SQL Server Agent Service is named SQLSERVERAGENT. If you take advantage of SQL Server 2000's multiple instance support, an additional named instance of the SQL Server service is created for each instance. The named instance of the SQL Server service will have the name MSSQL$InstanceName. Using either the Service Manager utility or the Services applet, you can stop, start, and pause each of the services as well as set their autostart property to automatically start each service when the server boots up.

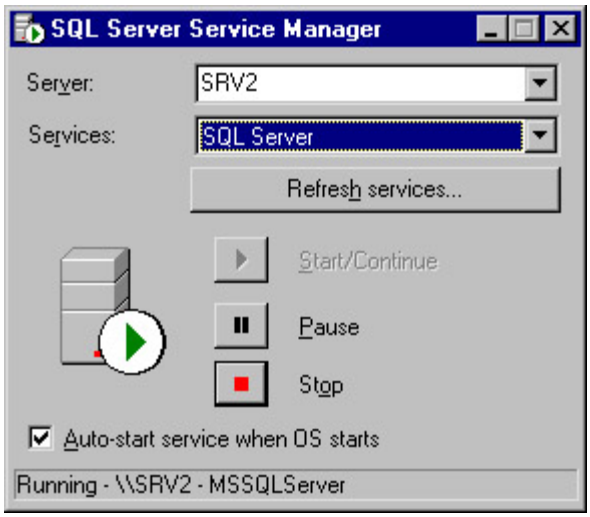


Figure 4.5: SQL Server Service Manager.

SQL Server Service

The SQL Server service is the database engine to SQL Server 2000. On Windows NT and 2000, SQL Server runs as a service. However, SQL Server Personal Edition can be run on Windows 95, 98, or Me and runs as a standard application due to lack of service support on these editions of Windows. Each installed instance of SQL Server has its own SQL Server service instance. The SQL Server service is responsible for all the primary tasks of SQL Server, including:

- Managing server resources and user connections
- Managing database files
- Executing T-SQL
- Locking management, keeping multiple users from updating the same data at the same time

Configuring the SQL Server service involves setting its startup user account (security context) and autostart property. In nearly all instances of SQL Server, the SQL Server service should be configured for autostart, which is the default installation setting. To review or modify the autostart status of your SQL Server service:

1. Open the SQL Server Enterprise Manager utility from the Start | Program Files | Microsoft SQL Server program group.
2. Expand the Microsoft SQL Servers | SQL Server Group icon to display the list of registered servers.
3. Right-click your server in the list, and select the Properties option from the context menu to display the SQL Server Properties (Configure) dialog box, as shown in Figure 4.6 .

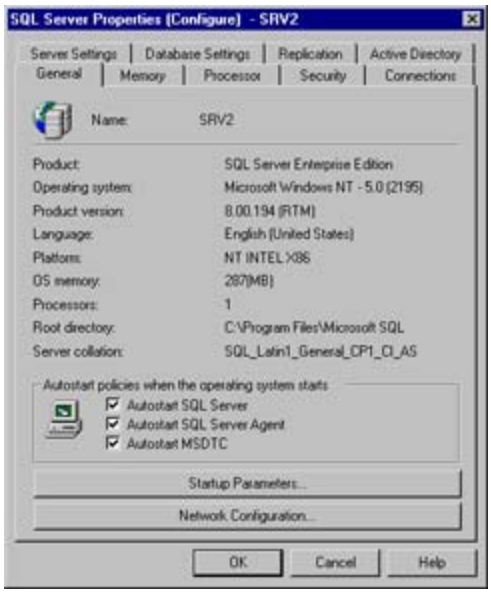


Figure 4.6: The SQL Server Properties (Configure) dialog box.

4. Review the Autostart SQL Server option in the "Autostart policies when the operating system starts" section. If the Autostart SQL Server option is checked, SQL Server is configured to autostart when the server boots up. Note that this option is also available from the Service Manager by selecting the "Autostart service when OS starts" check box for the SQL Server service.

From the SQL Server Properties (Configure) dialog box, you can also review the security context or startup user account for the SQL Server service. On the Security tab of the Properties dialog box, under the "Startup service account" section, the current security context of SQL Server is available. This property is initially set during the installation process but can be updated at any time using this dialog box. Two options are available for the service security account: System Account or This Account. The difference between the two options determines the kind of capabilities your SQL Server requires to communicate with other servers. The system account is a local security account that allows the SQL Server access to all local resources. The primary restriction here is that the system account has no network credentials, so communications with other servers is not possible. To allow your SQL Server full access to network resources and communications with other servers, you should configure your server with a domain user account, referred to as a *service account* .

The SQL Server service runs the sqlservr.exe file located in the Program Files\Microsoft SQL Server\MSSQL\Binn directory. You can view the amount of memory and processor time the SQL Server engine is consuming using the Windows Task Manager. To open the Windows Task Manager, right-click on an empty area of the Windows taskbar, and select Task Manager from the context menu. Select the Processes tab from the Task Manager, and locate the image name sqlservr.exe.

SQL Server Agent Service

SQL Server Agent service is responsible for scheduling and executing jobs as well as handling alerts and notifying operators of events that occur. Jobs in SQL Server are defined tasks that can execute one or more T-SQL statements. Jobs are typically scheduled to perform routine maintenance activities such as integrity checks and backup operations. Each multiple instance of SQL Server has its own copy of the SQL Server Agent service and uses the naming format SQLAgent$InstanceName.

In addition to scheduling and executing jobs, SQL Server Agent also manages alerts that are passed to it from the SQL Server service. SQL Server Agent can be configured to send e-mail, pages, or Internet send messages to specific users who are configured as operators in SQL Server. It is a common practice to configure database administrators as operators so that server errors such as low disk warnings or replication errors can be corrected immediately. You will learn how to configure these options in later chapters, but to review current SQL Server Agent settings now:

1. Open the SQL Server Enterprise Manager utility from the Start | Program Files | Microsoft SQL Server program group.
2. Expand the Microsoft SQL Servers | SQL Server Group icon to display the list of registered servers.
3. Select your server, and expand the Management folder to view the list of jobs, alerts, and operators that are currently configured.

You can configure many of the properties of the SQL Server Agent service from the SQL Server Agent Properties dialog box. To view the Properties dialog box, right-click the SQL Server Agent icon in the Management folder in Enterprise Manager, and select Properties from the context menu. From the Properties dialog box, you can review and modify the Startup service account as well as configure SQL Agent to restart the SQL Server service if it unexpectedly stops. Surprisingly, you cannot set the SQL Agent service's autostart option from the Properties dialog box. To view or change the autostart status of the SQL Agent, open the Service Manager utility, and review the "Autostart service when OS starts" check box for the SQL Server Agent service.

The SQL Server agent service runs the sqlagent.exe file located in the Program Files\Microsoft SQL Server\MSSQL\Binn directory. As with the SQL Server service, you can view the memory and processor time being used from the Task Manager utility.

Microsoft Distributed Transaction Coordinator Service

The MS DTC service allows applications to use other data sources on remote servers in a transaction. MS DTC is responsible for managing the transaction and either committing necessary changes on all servers or undoing any changes in the case of an error. SQL Server utilizes MS DTC to execute stored procedures on remote servers as well as performing updates on multiple OLE DB data sources.

You can configure the autostart property of the MS DTC service using the Service Manager utility. MS DTC is configured by default to autostart and should not be changed

unless you will not be using remote data sources in your SQL applications. The MS DTC service runs the msdtc.exe file located in the WINNT\ System32\ directory. As with the SQL Server service, you can view the memory and processor time being used by the MS DTC service from the Task Manager utility.

Microsoft Search Service

The Microsoft Search service is responsible for full-text indexing and executing full-text queries against SQL Server. If you have defined any full-text catalogs in your database, the Microsoft Search service is responsible for creating and updating those indexes. Full-text search requests are received by the Microsoft Search service for processing, and search results are returned.

You can configure the autostart property of the Microsoft Search service using the Service Manager utility. The Microsoft Search service runs the mssearch.exe file located in the Program Files\Common Files\System\ MSSearch\in\ directory. It is configured to use the local system account, which offers adequate permissions to complete its tasks. As with the SQL Server service, you can view the memory and processor time being used by the MS DTC service from the Task Manager utility.

MSSQLServerADHelper Service

The MSSQLServerADHelper service performs two functions that help integrate SQL Server with the Active Directory feature of Windows 2000. It adds and removes registered instances of SQL Server, SQL Server databases, publications, or analysis servers in Active Directory. It also ensures that the security account under which the SQL Server service is running has permission to update the Active Directory objects. The MSSQLServerADHelper service does not run all the time. It is started when needed and stopped when its task is completed. The MSSQLServerADHelper service cannot be managed using the Service Manager utility. If you need to review its configuration, you can use the Services applet from the Administrative Tools program group. The MSSQLServerADHelper uses a local system account for security and runs the sqladhlp.exe file located in the Program Files\Microsoft SQL Server\80\Tools\Binn\ directory.

MSSQLServerOLAPService

The MSSQLServerOLAPService is the Analysis Services component of SQL Server 2000. For information on the role of Analysis Services, refer to the Analysis Services chapter in this book.

Creating SQL Server Databases

As you would expect, before you can begin to physically lay out your database in SQL Server, you must design and plan your database solution. Planning your database is essential to its initial configuration and its future maintenance. There are four steps to delivering your basic database solution:

1. Design your database solution.
2. Design the physical database.
3. Create the database.
4. Monitor and maintain the database.

Designing Your Database Solution

The process of planning your database begins with your application and the problem that it will solve. Every database and software application has its roots in solving an existing business problem or meeting a new requirement. The business problem could be an existing application that needs to increase its scalability, or it could be an entirely new application for a new company, division, or market. Regardless of what has sparked the need for your new database solution, it begins with logically defining what your database will store and deliver to your applications. This planning phase will determines how your database solution will work—particularly the where, what, and how.

The "what" answer comes first. What your database solution will do largely entails the goals of your application and the information it will contain. Is your database solution an e-commerce product catalog and order-entry system, a knowledge base solution, or possibly an internal corporate application such as employee management? The answers to this question will allow you to estimate how much data will be in the database, what kind of distribution of the data is necessary, and which features of SQL Server are required to deliver your desired results. From this information, you can estimate the expected amount of database activity, such as table editing or bulk load operations. You can estimate the kind of lookup activity, such as full-text indexing.

All this information is necessary to answer the "where" and "how" questions regarding your database solution. Where will your database be located? The answer involves defining numerous configuration options, such as replication or partitioning of your database to support multiple remote locations, or high-availability requirements that represent "how" your database solution will work. Remember that greater performance and availability come at greater cost. Many midrange, multiserver hardware configurations can run into the hundreds of thousands of dollars in cost. This is a far stretch from a typical single-server configuration in the $10,000 to 50,000 range or even low-end servers in the $2,000 to 10,000 range. Balancing the needs of your application with its budget for it can be a trick in itself.

The process of procuring the hardware, software, and communications components necessary to complete your solution can begin by preparing a scaled-down development

environment. Very few organizations can afford to completely create a production-quality hardware environment before your application even begins to be developed. Even if your organization can, it would be misguiding and wasteful to do so. During the initial design and development stages, you can use existing development servers. Remember, many development projects can take months to complete, and more powerful and less expensive hardware is always around the corner.

You should also plan for change. As your application continues through design and into the development phase, your hardware or infrastructure requirements could change right along with it. It is pretty common to begin to procure hardware and necessary infrastructure components in the second half of the development phase. Why not wait until development is completed before setting up your production hardware environment? The answer is simple: You should always perform capacity or load tests against your production configuration in a controlled environment. The results of these tests will allow you to define measures that can be used to tune SQL Server and your application before it is released. You will always have to make adjustments to your equipment and software, but it is much more prudent to do so without interrupting actual users performing their jobs.

Database Modeling

After you have defined exactly what your database solution will do and how it will do it, you can begin to define a logical database model. Your logical database model will identify the information that will be collected and the organization of that information in your database. Designing your database is both an art and a science. Many of you have probably seen entity-relationship diagrams or heard of the process of normalization to create an efficient database model. A complete explanation of and the concepts behind designing your database application could easily consume a book of their own. Because a single section of a chapter is not enough to define this process, the latter portion of this chapter discusses some of the basic principles and tools available to assist you in designing your database application.

The importance of your database design for creating your physical database in SQL Server encompasses a high-level look at the information stored in your database and the activity in which it will be involved, such as updating and reading data. This logical design will assist you in designing your physical database. Although you could begin by simply creating a new SQL Server database—and many people do—you need to begin by conceptually understanding the data entities (logical groups of data) and the estimated read and write activity of those entities as well as their projected size and the potential growth of your database. This information will allow you to plan your data and log file structure, which is fundamental to creating your initial database in SQL Server.

Designing the Physical Database

From your logical database model, you can design your physical database structure. Your physical database design will include choosing the type of disk system to use and how to

lay out your database across the physical disks. Remember that databases are about storing and delivering data. All that information must be stored safely and be made available to users as quickly as possible. Selecting the appropriate disk system and laying out your database across that disk system using SQL Server's filegroups and files is essential to the performance and reliability of your database solution.

The Disk Subsystem

Recall from the first half of this chapter that SQL Server databases are made up of filegroups, data files, and log files. Each of these items can be used to effectively distribute your data across multiple physical disks to optimize read/write activity in your database. Too many highly active tables on one physical disk creates a bottleneck. Place your log file on a fault-tolerant disk system such as RAID5 and you could experience unnecessary performance hits. An intimate understanding of disk systems and some basic guidelines will get you on track to actually creating your database and preparing for development.

Nearly all product servers and many development servers in an organization typically employ some level of *fault tolerance* , especially in the disk system. What this basically means is that the server disk hardware is capable of withstanding a physical disk failure and continuing to operate with no data loss. This feat is accomplished using one of several levels of *redundant array of inexpensive/independent disks (RAID),* the term used to define a group of disks working together to physically store data. RAID has 11 recognized levels, each with its own benefits and drawbacks. Only four of these levels are commonly deployed. Each RAID level affects the number of disks required and the fault tolerance and performance of your disk configuration. Chapter 14 contains a detailed conversation on RAID, but let's quickly review the basic principles here so you are aware of their importance in the planning phase.

RAID employs multiple disks in an effort to increase system performance by spreading data across each disk. This process, called *striping* , increases performance by allowing each disk to locate and retrieve its portion of the requested data at the same time. Compared with a single disk that has a single read/write head that can move to only one area of the disk at a time, the benefit of striping is obvious. Consider asking your drive subsystem for a piece of data and have three or four drives retrieving it at once, and compare that to just one disk that has to move around the drive to collect all the pieces of the data you need.

The second aspect of RAID is fault tolerance. Fault tolerance in RAID configurations involves two methods. One is physically copying the entire data item to multiple disks so that each disk has the data. This process is called *mirroring* . In the event that one disk fails, the other disk already contains all the information and can continue to service any data requests. The second method of tolerance is a bit more complex; it employs the use of *parity information* . Parity information is calculated and written when data are stored to the disk, and the same parity data are verified when information is read from the disk. The use of parity data spread across multiple disks allows the data from a damaged disk

to be rebuilt. The end result of the various types of operations and their overhead produces different levels of read and write performance.

A review of the common RAID levels and their read/write performance appears in Table 4.2 and in Chapter 14 .

Table 4.2: RAID Levels and Performance

Level	Description	Fault Tolerance	Read Performance	Write Performance	Cost
No RAID	Single disk	No	Normal	Normal	Inexpensive
RAID0	Striping without parity	No	Fast	Fast	Expensive
RAID1	Mirroring	Yes	Normal	Normal	Moderate
RAID5	Striping with parity	Yes	Fast	Slower	Expensive
RAID0+1	Striping without parity with mirroring	Yes	Fast	Fast	Most expensive

Choosing the appropriate RAID level for your database files is important to the performance and reliability of your database solution. It is also essential to both budgeting and procuring database server hardware. Every database has both data files and log files that serve different purposes and undergo different levels of read and write activity. Log file write activity is *serial* , or sequential, so positioning log files on an isolated disk system maximizes write performance by reducing the amount of movement required by the read/write heads. Because log file activity is primarily write activity, choosing a disk system with RAID levels 0 or 0+1 is recommended. RAID0+1 is optimal, but many organizations can survive with reduced write performance and fault tolerance as well as the moderate cost level of RAID1. The choice of type of disk system to use is dependent on balancing your budget and meeting your fault-tolerance and read/write performance requirements.

The optimal disk configuration for your data files might require multiple RAID sets of different levels. First, you can eliminate any RAID levels that do not offer fault tolerance. Your database files should always be on fault-tolerant disks. Remember, your data is your business. From there, choosing the disk system depends on the type of activity your database will endure. For example, you might want to implement two separate RAID systems—one at RAID5 for data that are involved primarily in reporting or read-heavy activity and one at RAID1 for transaction data, which can be written and updated with frequency. Why not choose RAID0+1 from the start, because it seems to have both fast read and write activity and offers fault tolerance? As for cost, finances will always affect decisions, and the number of disks required to implement a complete RAID0+1 disk

system for the entire database could mean mixing multiple, less expensive systems tailored to specific portions of your database.

Capacity and Growth Planning

Now that you know what is involved in selecting a disk subsystem and which configurations are optimal for each of your database components, you need to determine just how much disk you will need. Capacity and growth planning are important to defining a configuration that is capable of accommodating your initial database and able to grow as your application grows. The process involves both exacting calculations and "guesstimates." At the beginning of your application, its storage requirements can be calculated with some level of certainty.

The future of your database application, however, depends on the success of your application or the subject matter behind it. This process largely involves examining business predictions of future customer and subject matter growth, such as new products and new markets. After you have determined record storage requirements for the entities in your database, you can use those numbers to estimate future storage requirements based on your business predictions of future growth. For example, if you know that each product record requires 1.2K of storage, then adding 1000 product records to the database will require 1.2MB of additional space. These numbers are relatively small, but if you expand this model across your entire database application, gigabytes of future storage space requirements will begin to appear.

In addition to normal table data storage, you must also determine the impact that additional data have on index storage requirements. As we reviewed in early sections, indexes are used throughout most databases to increase retrieval performance, but they also require additional storage space. The type of index (clustered or nonclustered) and the number of columns involved in the index correlate to the amount of space required. If you have numerous indexes with large numbers of columns, adding that 1.2K product record might actually require 1.8K of space overall to include the index records. In the following section, "Sizing Your Database," we review some example calculations to determine the exact impact of data additions.

Although many RAID configurations allow you to add disks without any application downtime, it is always better to plan for this change than to react to an "Out of disk space" error message on your server!

Sizing Your Database

Determining accurate estimates of your database size and the impact of additional data records is important to both the initial database configuration and future maintenance. Database sizing involves examining each table and its indexes to determine the physical storage requirements for each row. The process of calculating this result is simple once it is broken down into general steps. These steps involve:

- **Determine the number of rows** This value is simply the number of records that your database table will contain.
- **Determine the size of each data row** Calculating the row size of the data in your table requires identifying each fixed-length field and each variable-length field and determining the actual bytes required to store the row data. Two types of data types are available in SQL Server: fixed length and variable length. *Fixed-length* fields always require the same amount of physical storage for their contents. *Variable-length* column storage requirements are based on the amount of data stored in them. Unless you can accurately estimate the typical fill capacity of a variable-length column, you should estimate it to be 100 percent full. Refer to the SQL Server documentation for the rules on fixed- and variable-length fields, because particular variables have different storage requirements based on their use.
- **Calculate the number of data pages required** Each data page in SQL Server can store 8096 bytes of data. With this in mind, based on a single row's storage requirement, you can estimate the number of rows that will fit on a single data page. Row data cannot wrap to the next data page, so you need to round down to determine the exact number of rows per page. If your table contains a clustered index (which contains the data), you must adjust your calculation to include only the available amount of storage on each page, which equates to 8096 x (100-FILLFACTOR).

After you have calculated the number of data pages required to store your table, you can multiple that by 8096 to determine the actual storage space required for your table. The information you obtain from this process can also be used to plan for additional growth as new rows are added to your database.

During the physical design stage of your database, you can use estimates of the storage requirements based on numeric and character values being stored. Numeric values range from 1 to 4 bytes of storage, dependent on the variable type such as int, bigint, money, real, and so on. You can estimate character values by multiplying the number of characters being stored by 1 byte per character. Taking the total storage and adding an additional 10 percent will give you an estimate of storage requirements. This estimate does not take into consideration the impact of items such as unicode data, indexes, or fill-factor values. These estimates can be used to gain an idea of the storage capacity of entities for initial sizing estimates. You should not scale your database server storage to the initial database size estimates; SQL Server features such as file autogrow compensate for physical storage differences between data types and data page limitations that could result in a slightly larger database than your original estimates.

File Autogrow

Prior to SQL Server 7.0, managing database growth required constant monitoring and planning. This level of administration was due to database devices, the name used for storage units in SQL 6.5 and earlier, and their fixed size. Adding additional space to your databases required administrator intervention to expand the database device and the database.

Starting with SQL Server 7.0, Microsoft implemented *file autogrow* , which automatically increases the size of database or log files, when necessary, to accommodate additional information. A godsend for many database administrators, file autogrow can be configured to increase file growth at specified megabyte increments and percentage increments. Autogrow can also be restricted to a maximum file growth size or disabled to eliminate any file growth capabilities. You can review each of these options from the database properties dialog box, as depicted in Figure 4.7 for the Northwind sample database. The settings for the Northwind database are the default new database settings and are configured for file autogrow at 10 percent with unrestricted growth. This means that as the Northwind database reaches its full point, SQL Server will acquire an additional 10 percent of disk space to expand the database file. The Maximum file size | Unrestricted file growth setting causes this process to continue until no free space is available. You can also continue to adjust these settings as application database use changes. In the following section, "Creating and Configuring Databases," we will adjust the autogrow features of our example database.

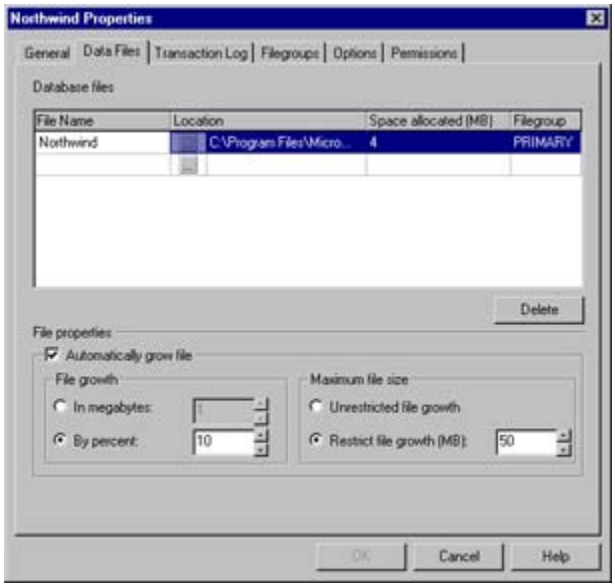


Figure 4.7: Northwind database File Autogrow Properties.

Creating and Configuring Your Database

Thanks to the continued usability enhancements of SQL Server, creating your physical database is the easiest part of getting started with your database solution.. SQL Server 2000 offers both graphical wizards and T-SQL statements that can be used to specify and create your database configuration. In the following sections, we step through each of these methods and examine the available options for defining new databases.

Before you can begin creating the actual database in SQL Server, you should have completed the following steps.

1. Identify your logical database model, including entities and projected read/write activity. This information allows you to define your physical disk configuration for optimal performance and reliability.
2. Calculate your initial database size and growth model. These numbers will allow you to set up your initial database and autogrow properties.
3. Define your database filegroups and files for physically storing your database information. After you have configured your initial database, you can configure the additional filegroups, data files, and log files for locating data across your physical disk systems.

Getting Started

In the following sections, we work through using the configuration principles you have developed so far to create a database in SQL Server 2000. For the purpose of our example, we will work on a database project for Southwind, the sister company of Northwind, which focuses on importing and exporting products between the United States and Central America. Based on the Northwind sample database application, we have identified the logical database model depicted in Figure 4.8 .

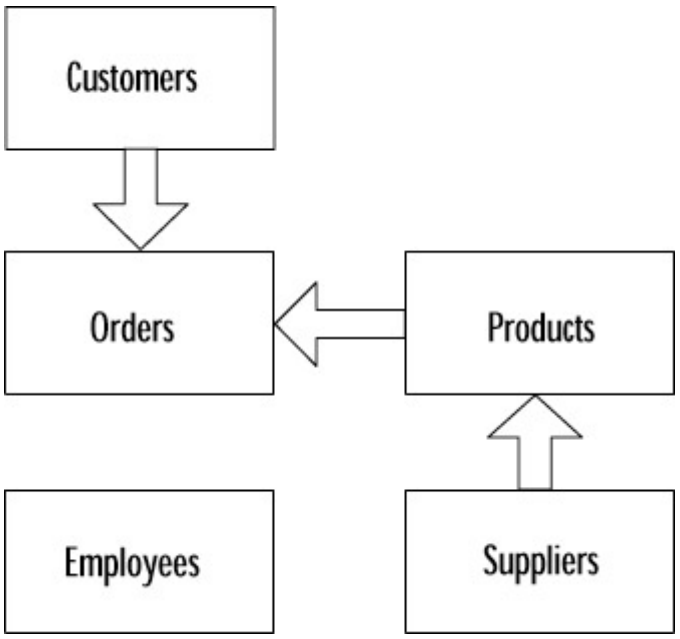


Figure 4.8: The Southwind logical database model.

Our logical database model has the properties outlined in Table 4.3 .

Table 4.3: Southwind Logical Database Properties

Entity	Read Activity	Write Activity	Initial Size w/ Index	Growth
Customers	Medium	High	5MB	High
Suppliers	Medium	Low	2MB	Low
Products	High	High	15MB	High
Orders	Medium	High	2MB	High
Employees	Medium	Low	1MB	Low

After reviewing the logical model, the entity activity, initial space requirements, and growth rate, we have created the physical database structure depicted in Table 4.4 for our SQL Server database. Based primarily on the read/write activity, we will create three filegroups to segment our database across three disk systems:

- **PRIMARY** Created as the default filegroup. It will contain the Customers and Suppliers entities.
- **FILEGROUP01** Will contain the Orders and Employees entities.
- **FILEGROUP02** Will contain the Products entity.

Table 4.4: Southwind Physical Database Properties

Property	Value
Database name	SOUTHWIND

Table 4.4: Southwind Physical Database Properties	
Property	**Value**
Filegroups	PRIMARY
	FILEGROUP01
	FILEGROUP02
Data files	SWINDDATA00
	Filegroup: PRIMARY
	Size: 8MB
	Autogrow: Enabled, 15%, Unrestricted
	Location: Disk1 (RAID1)
	SWINDDATA01
	Filegroup: FILEGROUP01
	Size: 3MB
	Autogrow: Enabled, 15%, Unrestricted
	Location: Disk2 (RAID1)
	SWINDDATA02
	Filegroup: FILEGROUP02
	Size: 15MB
	Autogrow: Enabled, 20%, Unrestricted
	Location: Disk3 (RAID0+1)
Log files	SWINDLOG00
	Autogrow: Enabled, 20%, Unrestricted
	Location: Disk4 (RAID1)

Each filegroup will contain a single data file located on a separate disk subsystem. The PRIMARY filegroup will be located on a RAID1 (mirrored) disk to balance cost with the average activity rates of the Customers and Suppliers entities. These entities are typically worked with by corporate staff, and the expense of RAID0+1 is above budget. The FILEGROUP01 filegroup will be located on a RAID1 disk—again, to balance cost with activity. The FILEGROUP02 filegroup will be located on a RAID0+1 disk. This decision is due to customer feedback that the speed at which customers can locate up-to-date product information is critical to their purchase. This means that Southwind must deliver constant product updates to the Products entity while customers continue to search and retrieve product information for purchasing. The single log file for our database will be located on a separate RAID1 disk to minimize read/write head movement and offer fault tolerance and moderate cost.

With all of that out of the way, you need to understand that this configuration is a bit complex to demonstrate the configuration options in SQL Server. We could get excellent

performance and fault tolerance out of a single RAID1+0 disk system or a RAID1 disk for the budget minded. The decision of grouping data entities and determining the appropriate disk system to use is dependent primarily on budget versus performance. The general rules are: Always use fault tolerance, and physically separate the data files from log files due to their disk access nature. The data sizes and growth rates in this example are estimates to reflect the individual configuration capabilities of each data file.

In the following steps, we implement our example database using both the Create Database Wizard and T-SQL to receive the same results.

Using the Create Database Wizard

The simplest method for creating your new SQL Server database is the Create Database Wizard. True to form, this graphical wizard will get even the novice up and running with a database solution in minutes. We walk through the database wizard to create our Southwind database.

Before you begin this example, unless you are fortunate enough to have a server with multiple RAID controllers available for your use to simulate our multiple disk systems, we need to create a directory structure to represent each disk so that we can locate our data and log files. Execute the following T-SQL script using Query Analyzer, or manually create the CH4 sample directories. This script uses the extended system-stored procedure to execute MAKE DIRECTORY (md) commands directly to the operating system console:

```
—Setup Create Database example directory structure to mimic
—multipled disk subsystems for filegroups, data files
—and log files placement.
USE master
EXEC dbo.xp_cmdshell 'MD "C:\SQL2K Examples\CH4\Disk1"'
EXEC dbo.xp_cmdshell 'MD "C:\SQL2K Examples\CH4\Disk2"'
EXEC dbo.xp_cmdshell 'MD "C:\SQL2K Examples\CH4\Disk3"'
EXEC dbo.xp_cmdshell 'MD "C:\SQL2K Examples\CH4\Disk4"'
```

Note This example assumes that your database server login is a member of the server administrators or database creators fixed server roles. Membership in one of these roles is required to create new databases in SQL Server. If you are working on your own test server and logging in as either the sa account or a trusted account that is part of the network administrators group, this will be true. If you are not sure, verify with your server administrator that your login account is a member of either of these fixed server roles.

The following sections (steps 1 through 21) review each step of the Create Database Wizard.

To begin:

1. Launch the Enterprise Manager utility from the Start | Programs | Microsoft SQL Server program group.

2. Expand the Microsoft SQL Servers | SQL Server Group icon, and locate the server on which you would like to create your new database.
3. From the Tools menu in Enterprise Manager, select the Wizards menu option to display the Select Wizard dialog box.
4. Expand the Database item by clicking the + icon.
5. Click the Create Database Wizard from the Database section to display the Create Database Wizard Welcome dialog box. The Welcome dialog box displays a list of the wizard steps:
    - Specify your database name.
    - Create one or more data files.
    - Specify data file autogrow properties.
    - Create one or more log files.
    - Specify log file autogrow properties.
6. Click the Next button to begin creating your database.

Create Database Wizard: Name the Database and Specify Its Location

The second wizard dialog box allows you to specify the database name and default data and log file locations:

7. Enter **Southwind** in the Database name: field. The database name can be up to 128 characters and must begin with a valid unicode letter, a to z or A to Z, as well as any other language letter character. The remaining characters of your database name can include @, #, _. The wizard allows you to enter invalid characters, but certain administrative functions will not work and could affect your server reliability. Microsoft recommends following the rules for identifiers for all database as well as other object names.
8. In the "Database file location:" field, click the Browse button (…) and locate the C:\SQL2K Examples\CH4\Disk1 folder we created using the script in Figure 4.8 . Our example directory will represent our first disk.
9. In the "Transaction Log file location:" field, click the Browse button (…) and locate the C:\SQL2K Examples\CH4\Disk4 folder we created using the script in Figure 4.8 . Our example directory will represent Disk 4.

Create Database Wizard: Name the Database Files

The third wizard dialog box allows you to specify multiple data files and their initial size. You cannot create additional filegroups at this point, so we will configure our first data file only:

10. In the Database files: list, select the Southwind_Data value and overwrite it with **SWINDDATA00** .
11. In the Initial Size (MB) field, enter **8** . To calculate the initial size value, we added together the initial size of the Customers and Suppliers entities from Table 4.3 and added an additional 15 percent to compensate for initial growth estimates. We estimated 15 percent based on moderate growth predictions for those entities

listed in Table 4.3 . With additional information such as each record size and projected number of new records, we could calculate the margin of growth, but for our example we will use our estimated autogrow value from Table 4.4 .

12. The SWINDDATA00 data file will be added to the PRIMARY filegroup. Using this wizard, we cannot specify additional filegroups, as we have outlined in our physical database plan. After we have completed the wizard, we will use the Database Properties dialog box to complete our filegroup and file configuration. After you complete your database configuration and review your database use, you will notice between 0.5MB and 1MB of space used. This space is used by system tables that are created in each new database to store configuration information. If you added any tables and data to your model database, this information will be automatically added to your new database. You might want to add additional space to your initial database size to compensate for the total use of system tables and additions to the model database. Now select the Next button to continue.

Create Database Wizard: Define the Database File Growth

The fourth wizard dialog box allows you to enable and disable file autogrow and specify autogrow properties for your data files:

13. In the "Grow the files by percent:" field, enter **15** to enable autogrow 15 percent. By default, autogrow is enabled for data files and set to 10 percent with unrestricted growth. We will not adjust the unrestricted setting, but in production databases you should always specify an upper limit to restrict the database files from completely filling the disk drive. If your data files run out of free space and can no longer grow, SQL Server will generate an 1105 error message.
14. Select the Next button to continue.

Create Database Wizard: Name the Transaction Log Files

The fifth wizard dialog box allows you to specify multiple log files and their initial size. You cannot create additional filegroups at this point, so we will configure our first data file only:

15. In the "Transaction log files:" list, select the Southwind_Log value and overwrite it with **SWINDLOG00** .
16. In the Initial Size (MB) field, enter **5** . We set our initial transaction log file to 5MB based on an overall write (Updates, Deletes, Inserts) activity level of medium and a total database size of 25MB, as depicted in Table 4.3 of our example database properties. The 5MB setting represents 20 percent of the total database size. In our example, we will perform daily transaction log backup that will result in the inactive segment of the transaction logs being truncated and space being reacquired for storage of new transactions. This is infrequent, on average, because many organizations perform hourly transaction log backups to increase their servers' recoverability. The inactive portion of the transaction log

contains the transactions (data changes) that have been physically committed to the database, so for SQL Server recovery, the transaction history is not needed in the transaction log. The transaction log backup contains the transaction data if a full database recovery is necessary in the future. (For a detailed description of SQL Server backup and recovery strategies, refer to Chapter 7 .) The important point here is that the initial size of your transaction log file is based on the expected write activity, which records transactions in the log, and the period between transaction log backups that free log space for reuse. You should always configure your transaction log for the expected maximum transaction log size and set a relatively large autogrow increment. This is necessary to avoid continued expansion of the transaction log, which can affect database performance. Now select the Next button to continue.

Create Database Wizard: Define the Transaction Log File Growth

As with our data files, the sixth wizard dialog box allows you specify the autogrow properties of the transaction log, allowing it to expand as needed to keep your database operational. You can also restrict the autogrow capabilities of the transaction log files:

17. In the "Grow the files by percent:" field, enter **20** to enable autogrow of 20 percent. By default, autogrow is enabled for log files and set to 10 percent with unrestricted growth. We will not adjust the unrestricted setting, but in production databases you should always specify an upper limit to restrict the file growth from completely filling the disk drive. You should configure the transaction log size so that autogrow should not be necessary, but in the event that log file autogrow is required, a large increment should provide the necessary space to reach the next backup point. If the transaction log has to continue to expand as a result of a small initial log file and low autogrow increment, your database performance will suffer.
18. Select the Next button to continue.

Create Database Wizard: Completing the Create Database Wizard

The final dialog box of the Create Database Wizard allows you to review the properties you have just configured and create your new database:

19. Select the Finish button to apply your settings and create your new Southwind database.
20. Select OK from the status dialog box stating that your database was successfully created.
21. The next prompt will ask you if you would like to create a maintenance plan for the Southwind database. We will use the Maintenance Plan Wizard to complete this step later in this example, so you can select the No button to continue. The wizard displayed if you had selected Yes is the same one as we will manually launch later to create our maintenance plan.

Configuring Your Database

As we worked through the Create Database Wizard, you should have noticed that we were never asked to create our filegroups, which we defined in our example database configuration. This limitation reflects the goal of the wizard: to get novice administrators up and running with new databases as quickly as possible. Microsoft recommends using a single database file and a single transaction log file for simple database applications, which, again, reflects the expected use of this wizard. We are not limited, however, in creating our desired configuration quickly and easily using other graphical tools in SQL Server.

In the following example, we will step through the Database Properties dialog box to complete the configuration of our database and review each of the available configuration settings. To display the database properties dialog box:

1. In the Enterprise Manager utility, expand the Microsoft SQL Servers | SQL Server Group icons, and select the server on which you created the Southwind database.
2. Expand the server icon to display the list of SQL Server object folders.
3. Double-click the Databases folder to display a list of databases for our test server.
4. Right-click the Southwind database icon, and select Properties from the context menu to display the Southwind Properties dialog box shown in Figure 4.9 .

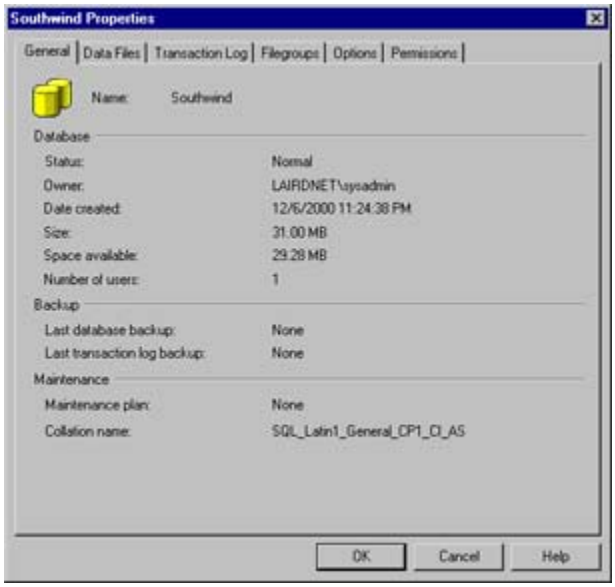


Figure 4.9: The General tab of the Southwind Properties dialog box.

As Figure 4.9 displays, the Database Properties dialog opens to the General tab, displaying the general database properties, including:

Name The database name we specified during the Create Database Wizard. Although it is not recommended that you modify the name property after you begin creating your database application, you can use the system-stored procedure sp_renamedb to rename

your database. For example, to rename our Southwind database to Southwind2, we can execute the following T-SQL statement in the SQL Query Analyzer:

```
EXEC sp_renamedb 'Southwind', 'Southwind2'
```

Database | Status The status reflects the current configuration state of the database. This value will typically be set to Normal but could be set to Standby or Offline, given different configuration settings. For example, if you set the database access option on the Options tab to read-only, the status will be set to standby. For complete status details, you can use the system-stored procedure sp_helpdb. Execute the following T-SQL statement in the SQL Query Analyzer utility and review the status column for complete status settings:

```
EXEC sp_helpdb 'Southwind'
```

Database | Owner The current owner of the database. This is typically the user account that created the database. To create a new database, you must be a member of the system administrators (sysadmin) or database creators (dbcreator) fixed server roles. If you are the current database owner or a member of the system administrators role, you can execute the following T-SQL statement in the SQL Query Analyzer utility to change the database owner to the sa user:

```
USE Southwind
EXEC sp_changedbowner sa
```

Database | Date created The date the database was created. This value is stored in the sysdatabases system table, crdate field located in the master database. It is recommended that you do *not* directly query system tables or update them, but you can review this setting by executing the following T-SQL statements in SQL Query Analyzer:

```
USE master
SELECT crdate FROM sysdatabases WHERE name='Southwind'
```

Database | Size The total space allocated to the database. This includes data files and transaction log files. This value is 13MB for our Southwind database—8MB for our initial data file plus 5MB for our transaction log file.

Database | Space available The total remaining space in all data files and transaction log files. Although we have added no objects to our database yet, the Southwind database already shows approximately 1MB of used space. This used space is for system tables that are created in every new database to record configuration information.

Database | Number of users The total number of users defined in the database. You can review the list of users by selecting the Users object group in Enterprise Manager for the Southwind database. This information is also stored in the Sysusers system table located in the Southwind database.

Backup | Last database backup The date and time of the last database backup event.

Backup | Last transaction log backup The date and time of the last transaction log backup event.

Maintenance | Maintenance plan The name of the configured maintenance plan for this database. We will configure a maintenance plan later in this chapter to verify the integrity of our database and execute backups.

Maintenance | Collation name The database collation setting specifies the language-specific settings for the database. The collation is not a maintenance option; this is one example of "Oops, no place else to put it" control placement. New to SQL Server 2000, each database can have its own collation setting. You can modify the default collation setting for your database using the ALTER DATABASE … COLLATE statement. Changing the default collation does not adjust the collation setting for existing objects, though. The new setting applies only to new objects created in the database after the setting was modified. To update existing tables, you need to execute the ALTER TABLE statement for each table in the database to specify the new collation setting. With this in mind, you should not adjust collation settings after your database development has begun. In fact, to modify the default collation setting for SQL Server, you need to bulk-copy export all data from your user databases, drop all user databases, rebuild the master database with the new collation, recreate all database objects, and bulk-load import the data. That sounds like a lot of work, and it is—so be sure to choose the appropriate collation setting ahead of time!

Southwind Properties | Filegroups

After you have completed reviewing the general information section of the Database Properties dialog box, we can continue configuring our Southwind database by creating our additional filegroups and data files. Microsoft states that most databases can successfully use a single filegroup and transaction log file, and that opinion is obvious in the order of the tabs in the Properties dialog box and the process of creating a new filegroup. Although the Filegroups tab is fourth in order, we will use it first to create the additional filegroups for our example database. If we do not create additional filegroups before we create new data files, we will not be able to properly assign them to their filegroups, and we'll have to delete the data files and recreate them after our filegroups. This process sounds a bit inefficient, and it is—and it makes a strong case for the T-SQL CREATE DATABASE statement we will use later to create this entire database configuration using one long T-SQL statement.

To create our remaining two filegroups using the Properties dialog box:

1. Click the Filegroups tab of the Southwind Properties dialog box, then click in the first empty name field in the Files list.
2. Enter **FILEGROUP01** in the name field. The remaining properties cannot be changed at this point.

3. Click the empty name field below the FILEGROUP01 row, and enter **FILEGROUP02** in the name field. The list should now show PRIMARY, FILEGROUP01, FILEGROUP02 as valid filegroups.
4. Because there is no Apply feature of this dialog box, in order to save your filegroup additions and use them with your new data files, you will have to select the OK button to close the Properties dialog box, and then manually reopen it. After the Properties dialog box has closed, right-click the Southwind database in Enterprise Manager, and select the Properties option from the context menu to display the Southwind Properties dialog box again.
5. Next, click the Filegroups tab, and verify that our new filegroups are listed there. Both FILEGROUP01 and FILEGROUP02 will have 0 listed under the Files column, and you will not be able to check the Read-only or Default options until after you have assigned new data files to them.

Southwind Properties | Data Files

Now that we have created our additional filegroups to better place our database objects across our various disks, we can create our remaining data files. Click the Data Files tab to view the SWINDDATA00 data file we created using the Create Database Wizard. The SWINDDATA00 data file is listed with the previously specified location and space allocated. The filegroup property has been automatically set to PRIMARY. Recall from our conversation at the beginning of the chapter that the PRIMARY filegroup is created automatically and is the default filegroup for objects in your database. What that means is that if you create a new table or index and do not specify a filegroup, the object will be placed in the PRIMARY filegroup. With this in mind, if you use multiple filegroups in your database, you must remember to properly allocate your database objects to those filegroups in order to gain the advantages of data and activity distribution across your disk subsystems.

If you select the SWINDDATA00 file, the autogrow properties in the bottom half of the dialog box will change to display the configured settings for that particular data file. Each data file can contain its own autogrow properties.

To continue creating our multiple data file configuration:

1. On the Data Files tab of the Southwind Properties dialog box, click in the first empty File Name field and enter **SWINDDATA01** . As you begin to enter this value, the Location, Space Allocated, and Filegroup values will automatically fill in with defaults.
2. In the Location field, click the Browse button (…) to display the Locate Database File dialog box, as shown in Figure 4.10 .

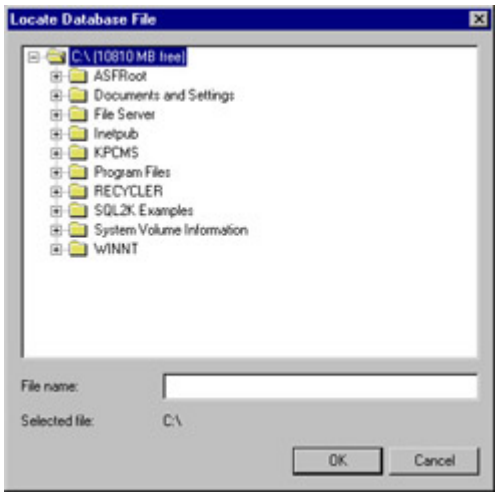


Figure 4.10: The Locate Database File dialog box.

3. In the Locate Database File dialog box, select the C:\SQL2K Examples\CH4\Disk2 folder in the folder tree to place our new data file on our second example disk.
4. In the "File name:" field, change the value to SWINDDATA01.NDF.
5. Click the OK button to set our location properties and return to the Southwind Properties dialog box.
6. In the "Space allocated" field, enter **4** , since 4MB represents the initial size total of our Customers and Employees entities plus the initial growth rate of 15 percent, which equals 0.45MB. Because we cannot enter fractional MB sizes, we rounded up to 4MB.
7. In the Filegroup field, select the drop-down list and select our new filegroup, FILEGROUP01.
8. To create your third data file, repeat Steps 1 through 7 with these properties:
    - File Name: **SWINDDATA02**
    - Location: C:\SQL2K Examples\CH4\Disk3\SWINDDATA02.NDF
    - Space allocated: **18**
    - Filegroup: FILEGROUP02
9. To save your new data file settings, click the OK button to close the Properties dialog box. After the Properties dialog box has closed, right-click the Southwind database in Enterprise Manager and select the Properties option from the context menu to display the Southwind Properties dialog box again.
10. If you return to the Filegroups tab on the Properties dialog box, you will find that the Read-only and Default options are now available for either of our new filegroups. If you enable Read-only for a filegroup, it cannot be modified in any way; this includes adding new objects to it. The default property can be specified for only one filegroup at a time and specifies the default location for objects created in your database.

Southwind Properties | Transaction Log

The Transaction Log tab of the Southwind Properties dialog box lists the transaction log files that have been defined for our database. One of the obvious differences here is that there is no filegroup assignment for log files. You can create up to 32,767 log files (the number of data files) for your database, but you cannot specify groups of log files or their write order. Each log file will be written to until it is full, and then the next log file will be used, and so on. This process explains our choice of placing our log file on an isolated, mirrored disk system that offers fault tolerance and balances cost against write performance. Because this disk will incur only log-writing activity, the read/write head of the disk drives will sequentially move across the disk, providing optimal performance. If our disk system runs low on space, we can add log files on other disks to accommodate the growing demands. You could also perform log file backups more frequently to free transaction log space and maintain a healthy log file size. The choice of approach you follow is dependent on your hardware capabilities and database activity.

For our example database, we will configure only a single transaction log file, SWINDLOG00.LDF. You can use the Transaction Log tab to add transaction log files or adjust autogrow settings at any time.

Southwind Properties | Options

The Options tab of the properties dialog box provides access to many of the configuration settings for your database. You can set the access properties, recovery, database settings, and compatibility level for the Southwind database. The default settings here are obtained from the model database settings at the time the new database is created. The following list explains each of these settings in detail:

- **Access | Restrict Access** In SQL Server, you can control overall access to your database by enabling restricted access and specifying the access level. Restricted access can be given to only the members of the db_owner, dbcreator, or sysadmin fixed server roles or to only a single user at a time. Although not of much value in common application scenarios, restricted access can be set for administrative purposes such as temporarily disabling access during maintenance.
- **Access | Read-only** To disable any modifications to objects or data in your database, you can enable the read-only access property. If read-only is enabled, data can only be retrieved. No locking occurs, which can speed up query performance. Read-only databases are commonly used in reporting scenarios and distributing information such as product catalogs to remote applications.
- **Recovery | Model** The Recovery Model property of your database determines the backup capabilities and potential data loss during recovery of your database. For more detail on the effects of the recovery model to backup and restore operations, please see Chapter 7 . Three options exist for the recovery model setting:
    - **Simple** *Simple recovery* is similar to the trunc. log on chkpt. setting in previous versions of SQL Server. In simple recovery mode, the transaction log is truncated each time a checkpoint event occurs. The checkpoint event is when transaction log changes are physically written to the database. For

this reason, your database can be recovered to only the most recent full or differential database backup. This setting is not recommended for production databases unless data loss between backups is acceptable. Development servers can be configured for simple recovery to maintain small transaction log files and because data loss is inconsequential.
    - **Full** *Full recovery* provides complete transaction logs for backup, allowing you to recover your database to the most recent transaction log backup or any point in time. All transactions, including bulk-copy operations, are fully logged. This should be your production database recovery setting.
    - **Bulk-logged** *Bulk-logged recovery* is similar to the full recovery model except that certain operations are minimally logged. These operations include SELECT INTO, BULK INSERT, bcp, CREATE INDEX, and text and image operations, including WRITETEXT and UPDATETEXT. The database is recoverable to the end of the last transaction log if it contains bulk operations, not to any point in time, as is available with the full recovery model. This limitation is due to the minimal logging aspect of bulk operations, which results in the inability to control these activities on an operation-by-operation basis.
- **Settings | ANSI NULL default** This setting specifies whether column values should be set to NULL or NOT NULL by default.
- **Settings | Recursive triggers** Enabling recursive triggers allows for direct recursion or the ability for a trigger to fire, performing an operation on its own table, which causes the same trigger to fire again. This setting does *not* affect indirect recursion, the ability for a trigger to fire, which operates on a different table, causing that trigger to fire, which then operates on the original table, causing the original trigger to fire again.
- **Settings | Auto update statistics** Statistics are used by the query optimizer to determine the most efficient method of performing an operation such as joining multiple tables. The "auto update statistics" setting is enabled by default based on the model database setting. This setting allows out-of-date statistics caused by table changes to be updated automatically during optimization.
- **Settings | Torn page detection** This setting enables the detection of torn pages. *Torn pages* are write processes that were interrupted before they were completed, typically due to hardware failure or, for example, a battery failure on a notebook during a page write operation. Torn pages can lead to data loss. This setting is enabled by default and should not be modified.
- **Settings | Auto close** The auto-close property determines if the database is "closed" after all users have disconnected and resources have been freed. The database is not "opened" again until a user connection request is received. By default, this property is not enabled except in the Desktop Engine edition of SQL Server (MSDE). In dedicated SQL Server installations, auto close is not necessary and leads to minor overhead for the close and open operations. In MSDE applications, the auto-close property allows data files to be copied or managed in the same manner as normal files by releasing them when there is no activity. If the

auto-close property were disabled in your MSDE database and you attempted to copy the database file, you would receive a "file in use" error message.

- **Settings | Auto shrink** The auto-shrink property determines whether or not SQL Server can automatically shrink a database file that contains more than 25 percent of free space. Database files are shrunk to a size that results in 25 percent of free space. This option is not enabled by default in SQL Server, with the exception of MSDE.
- **Settings | Auto-create statistics** If auto-create statistics is enabled, statistics are automatically created for columns to improve query performance. This option is enabled by default in SQL Server.
- **Settings | Use quoted identifiers** The quoted identifiers property allows you to delimit identifiers with double quotation marks. Quoted identifiers do not need to conform to the rules for identifiers and could contain reserved characters. This option is not enabled by default.
- **Compatibility | Level** SQL Server 2000 supports setting the database compatibility level to remain compatible with previous versions of SQL Server. *Compatibility* refers to reserved words and T-SQL extensions that might have been modified or removed from later versions of T-SQL. If you have a database application that utilizes features specific to a particular version, you can set the database compatibility level so that the application can continue to work on SQL Server 2000. Valid compatibility levels are:
    - **60** SQL Server 6.0 behavior compatibility
    - **65** SQL Server 6.5 behavior compatibility
    - **70** SQL Server 7.0 behavior compatibility
    - **80** SQL Server 2000 behavior compatibility
- **Active Directory | List this database in Active Directory** New to SQL Server 2000 is the ability to register SQL Servers, publications, data-bases, and Analysis Servers in Windows 2000 Active Directory. This level of directory integration offers location independence throughout the enterprise, allowing database servers and objects to be located with existing directory search tools. If your SQL Server is enabled for Active Directory, you can enable the List This Database option for your data-base and include its listing in Active Directory.

Southwind Properties | Permissions

The final tab in the Database Properties dialog box determines the create object permissions level of database users and roles in your database. Because we have not modeled our security for the database yet, we will not modify the Permissions tab properties. You can assign several permissions that allow or disallow users and roles the ability to create objects in your database and perform database and transaction log backups. If you click the permission once, you will enable the permission for the user or role, and the check box will be filled with a green check mark. If you click the check box a second time, it will contain a red X, signifying that you have removed that user's or role's ability to perform the desired function. The valid permissions are:

- **Create Table** Create new table objects in your database.

- **Create View** Create new view objects in your database.
- **Create SP** Create new stored procedures in your database.
- **Create Default** Create new defaults in your database.
- **Create Rule** Create new rules in your database.
- **Create Function** Create new user-defined functions in your database.
- **Backup DB** Perform database backup operations on your database.
- **Backup Log** Perform transaction log backup operations on your database.

Reviewing the Southwind Configuration

After you have completed all these steps. you are ready to begin creating your actual database application, including your database tables, views, stored procedures, users, and assigning user permissions. Many more wizards and T-SQL commands are available to accomplish these tasks. In the last sections of this chapter, we review some of the design tools available for modeling your actual entity-relationship diagrams and creating tables, views, and other objects in your database.

To view your physical database files, you can navigate your C: drive to the SQL2K Examples\CH4 directory and open each disk folder to locate the files that make up your database. If we had placed each one of these files on its own disk subsystem, we would experience the optimal performance for which we designed our database. As your application grows, you might want to add additional data files to accommodate space requirements and achieve optimal performance. This process will continue throughout the life of your application. In the following example, we bypass all the graphical tools we used to create our database and write a single T-SQL script to accomplish the identical task.

Using T-SQL to Create and Alter a Database

In the previous section , we used the Create Database Wizard and reviewed all the configuration settings available from the Database Properties dialog box. Due to the basic limitations of the Create Database Wizard, we had to use two different processes to achieve our desired database configuration. On top of that, we had to save and reopen the Database Properties dialog box several times to save our changes and continue with our configuration. All of that sounds like a lot of work and is a bit inefficient for experienced database administrators. As with nearly all the graphical functions in SQL Server Enterprise Manager, T-SQL equivalents will allow you to escape all the prompts and check boxes and create your database with a single script.

The following T-SQL example first deletes the Southwind database we created in the previous example and then recreates it using the T-SQL CREATE DATABASE statement. The new database will be identical to our previous example. You can execute this script using the Query Analyzer tool located in the Programs | Microsoft SQL Server program group:

```
—Delete the existing Southwind database if it exists.
—If you have the database open, a database in use error will occur.
```

```
—Exit the Enterprise Manager utility to disconnect
—from the Southwind database so that it can be deleted.
IF EXISTS (SELECT name FROM master.dbo.sysdatabases
WHERE name = N'Southwind')
  DROP DATABASE [Southwind]
GO

—Create new Southwind database according to our physical design
—creating each filegroup and placing each data file on a separate




—example disk
CREATE DATABASE Southwind
ON PRIMARY
( NAME = SWINDDATA00,
  FILENAME = 'C:\SQL2K Examples\CH4\Disk1\SWINDDATA00.mdf',
  SIZE = 8,
  MAXSIZE = UNLIMITED,
  FILEGROWTH = 15% ),
FILEGROUP FILEGROUP01
( NAME = SWINDDATA01,
  FILENAME = 'C:\SQL2K Examples\CH4\Disk2\SWINDDATA01.ndf',
  SIZE = 3,
  MAXSIZE = UNLIMITED,
  FILEGROWTH = 15% ),
FILEGROUP FILEGROUP02
( NAME = SWINDDATA02,
  FILENAME = 'C:\SQL2K Examples\CH4\Disk3\SWINDDATA02.ndf',
  SIZE = 15,
  MAXSIZE = UNLIMITED,
  FILEGROWTH = 20% )
LOG ON
( NAME = 'SWINDLOG00',
  FILENAME = 'C:\SQL2K Examples\CH4\Disk4\SWINDLOG00.ldf',
  SIZE = 5MB,
  MAXSIZE = UNLIMITED,
  FILEGROWTH = 20% )
GO

Results:
Deleting database file 'C:\SQL2K Examples\CH4\Disk3\SWINDDATA02.ndf'.
Deleting database file 'C:\SQL2K Examples\CH4\Disk2\SWINDDATA01.ndf'.
Deleting database file 'C:\SQL2K Examples\CH4\Disk4\SWINDLOG00.ldf'.
Deleting database file 'C:\SQL2K Examples\CH4\Disk1\SWINDDATA00.mdf'.
The CREATE DATABASE process is allocating 8.00 MB on disk
'SWINDDATA00'.
The CREATE DATABASE process is allocating 3.00 MB on disk
'SWINDDATA01'.
The CREATE DATABASE process is allocating 15.00 MB on disk
'SWINDDATA02'.
The CREATE DATABASE process is allocating 5.00 MB on disk 'SWINDLOG00'.
```

From the results, we can see that the DROP DATABASE command removes the database from SQL Server and deletes all associated data and log files. The second

statement, CREATE DATABASE, created our Southwind database, complete with the filegroups, data files, and log files that we outlined in our example. To verify that your new database exists:

1. Open the Enterprise Manager utility and navigate to your server.
2. If the Southwind database is not visible, right-click the Databases folder under your server, and select the Refresh menu option.

Through this example, you can easily see the powerful use of T-SQL commands for managing your database objects. You can save this script and re-execute it any time on any database server to recreate the Southwind database. If you use the wizards, you must go through each step to complete the task, and there is no guarantee that you will not miss something along the way. To be able to recreate your database exactly as you designed it, you should always create a script file to accomplish the task instead of using the wizards. You can easily back up your scripts and have them available in the event that you need to rebuild your database server.

In addition to the CREATE DATABASE statement, you can make changes to your database using the ALTER DATABASE statement. For example, to add an additional data file to filegroup FILEGROUP02, execute the following statement using the SQL Query Analyzer:

```
ALTER DATABASE Southwind
ADD FILE
( NAME = SWINDDATA03,
 FILENAME = 'C:\SQL2K Examples\CH4\Disk3\SWINDDATA03.ndf',
 SIZE = 5MB,
 MAXSIZE = 25MB,
 FILEGROWTH = 2MB
)
TO FILEGROUP FILEGROUP02
GO
Result:
Extending database by 5.00 MB on disk 'SWINDDATA03'.
```

If you return the Database Properties dialog box in Enterprise Manager, you can verify that an additional data file has been added to FILEGROUP02. You can also use the ALTER DATABASE statement to set all the available database options that we reviewed in the previous Database Properties dialog. For example, to set the recovery model for our database to Full, execute the following command in the SQL Query Analyzer tool:

```
ALTER DATABASE Southwind
SET RECOVERY FULL
```

The capabilities of T-SQL commands go well beyond those of the graphical wizards and dialogs. Wizards and dialogs are excellent learning tools for new administrators, but you will quickly find yourself writing T-SQL scripts to automate and control your database servers. For a complete explanation of all the options available using the ALTER DATABASE statement, please refer to the SQL Server Books Online.

Monitoring and Maintenance

SQL Server 2000 offers a wide array of monitoring and maintenance tools, including Performance Monitor counters, backup and restore utilities, Database Console Commands (DBCC), and system-stored procedures. Many of these options are discussed in detail in later chapters of this book. For new database administrators, a wizard is included in SQL Server 2000 to get you up and running with a basic maintenance and backup plan, complete with logging, so you can monitor its activity.

Database Maintenance Plan Wizard

Recall from our previous example using the Create Database Wizard that when we completed the wizard, we were asked to create a new maintenance plan for our database. If we had answered Yes to that prompt, SQL Server would have started the Maintenance Plan Wizard and walked us through the process of completing this task. As with all the SQL Server wizards, we can still access the Maintenance Plan Wizard from the Tools | Wizards menu in Enterprise Manager. In the following sections (steps 1 through 43), we create for our database a basic maintenance plan that will check for data integrity and perform weekly database backups and daily transaction log backups.

To begin:

1. Open the Enterprise Manager utility, and navigate down to select our database server.
2. From the Tools menu, select the Wizards menu option to display the Select Wizard dialog box.
3. Expand the Management wizard group, and click the Database Maintenance Plan Wizard option.
4. Select the OK button to launch the Database Maintenance Plan Wizard Welcome dialog box, outlining the tasks we will complete using the wizard.
5. Select the Next button to display the Select Databases step.

In the following sections, we walk through each of the Maintenance Plan Wizard dialog boxes and create our maintenance plan to keep our database in good health.

Maintenance Plan Wizard: Select Databases

The Select Databases step allows you to create your maintenance plan for multiple databases at once. You can choose to apply your maintenance plan to all databases, to only system databases (including msdb, Master, and Model), to all user databases only (all databases except system databases), or to only the databases we select. In Enterprise Edition of SQL Server 2000, following the list of databases, an additional option called log shipping is available. Log shipping is an advanced configuration option that transfers transaction log information to a second server to be applied. This option creates a backup server with near-exact data values; this server is called a warm standby server. Chapter 3 in this book delves into the topic of log-shipping configuration and capabilities.

For our example, we will create a maintenance plan for our new Southwind database only:

6. Select the These Databases option, and check the Southwind database in the Databases list.
7. Select the Next button to continue.

Maintenance Plan Wizard: Update Data Optimization Information

Your maintenance plan can include rebuilding indexes, updating statistics, and reorganizing free space to provide more efficient query performance and data storage. You can select from the following options:

Reorganize data and index pages If you select this option, your indexes will be rebuilt based on the schedule you set. Rebuilding the indexes will rearrange the data and provide space for additional data on each page. To determine how much space should be left on each page for new data, you can select whether to use the original FILLFACTOR setting or specify a new fee space percentage. The FILLFACTOR property can be specified when you define a new index, and it specifies how full the page should be when the index is created. The FILLFACTOR is not maintained as data are added, however, so your data pages can fill to capacity, requiring page splits as data are added to full pages. Page splits are time consuming and can affect performance, so maintaining healthy free-space percentages is important to your database performance. We will reorganize our data to provide healthy fill levels:

8. Check the “Reorganize data and index pages” check box.
9. Select the “Reorganize pages with original amount of free space” radio button.

Update statistics used by query optimizer If you elect to rebuild your index pages as the first option, this option will be disabled because rebuilding your indexes automatically updates statistics. If reorganizing your data is not desired, you can update statistics by checking this option. Statistics are used for optimization by the query processor and can lead to performance improvements in your T-SQL statements. You can also adjust the percentage of sampling to be used in creating new statistics. The percentage you enter is the percentage of the table data that should be used to create new statistics. A larger percentage will create more accurate statistics but will also take longer.

Remove unused space from database files If you have a large amount of activity, your database files can grow excessively to support the additional data but will remain that maximum size even after data have been removed. To provide more efficient allocation and file management, you can periodically reduce the size of your files. Selecting this check box enables resize checking according to the trigger size and free space requested. For our database, we will enable this check and specify a checking size of 100MB so that the maintenance plan will determine if it is possible to shrink the database after it exceeds

100MB. The free space percentage of 10 percent is acceptable, so we will not adjust that setting:

10. Check the “Remove unused space from database files” check box.
11. Overwrite the “When it grows beyond” setting to **100** .

Schedule We can specify a schedule to run these maintenance activities. The SQL Agent service will monitor for our scheduled event and execute it independently of the remaining choices in our maintenance plan. For example, you might want to update statistics every other day at 2:00 am so that user activity will not be affected:

12. Click the Change button to display the Edit Recurring Job Schedule dialog box shown in Figure 4.11 .

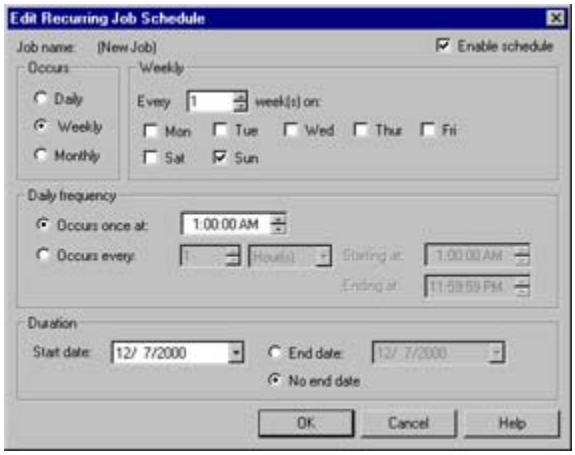


Figure 4.11: The Edit Recurring Job Schedule dialog box.

13. Set the Recurring Job Schedule to the following values:
    a. Occurs: Daily
    b. Daily: Every 2 Days
    c. Daily frequency: Occurs once at 2:00:00 am
14. Select the OK button to save our new schedule and return to the wizard.
15. Select the Next button to continue creating our maintenance plan.

Maintenance Plan Wizard: Database Integrity Check

Periodically scanning your database for object integrity and allocation is an important proactive task that can prevent minor problems from becoming major ones. Many administrators periodically run the DBCC CHECKDB to perform these checks. The maintenance wizard will assist you in setting up scheduled execution of the DBCC CHECKDB statement on your database:

Check database integrity Selecting the “Check database integrity” option will run the DBCC CHECKDB command on your database at the scheduled interval that you set. The DBCC CHECKDB verifies the integrity of index and data pages. You can also instruct

the DBCC CHECKDB command to correct any errors that it finds. If you select the "Attempt to repair any minor problems" check box, the database will be set to single-user mode while the maintenance job runs. With this in mind, you should schedule integrity checks for off-hours so that user activity is not affected.

Perform these checks before doing backups In addition to regularly scheduling integrity checks on your database, you can specify to have the integrity checks executed before database backups. If the integrity checks return errors, the backup operations will not continue. For our database, we will enable integrity checks and specify that problems should be corrected:

16. Check the "Check database integrity" check box and select the Include Indexes option to perform a full integrity check on our database.
17. Check the "Attempt to repair minor problems" check box to allow the DBCC CHECKDB process to fix any integrity problems that it encounters.
18. Click the Change button to display the Edit Recurring Job Schedule dialog box (refer back to Figure 4.11).
19. Set the Recurring Job Schedule to the following values for daily checking:
    a. Occurs: Daily
    b. Daily: Every 1 Days
    c. Daily frequency: Occurs once at 2:30:00 am
20. Select the OK button to save our new schedule and return to the wizard.
21. Select the Next button to continue creating our maintenance plan.

Maintenance Plan Wizard: Specify the Database Backup Plan

The next step in the wizard allows us to set a full backup schedule to be included in our maintenance plan. A discussion of designing and building a back-up strategy is outside the scope of this chapter; you should refer to Chapter 7 for the options and details in SQL Server 2000. For the purpose of our example, we will set up a weekly backup to disk and verify the backup's integrity to ensure that a valid backup has taken place:

22. Check the "Backup the database as part of the maintenance plan" check box.
23. Check the "Verify the integrity of the backup when complete" check box.
24. Verify that "Location to store the backup file to" is set to Disk.
25. The schedule is set to Weekly on Sunday at 2:00 am by default, which we will use for our example. If you want to modify this schedule, you can select the Change button and adjust the schedule accordingly.
26. Select the Next button to continue creating our maintenance plan.

Maintenance Plan Wizard: Specify the Backup Disk Directory

Because in the previous step we elected to back up our database to disk, the wizard will now prompt you for the location to store the backup files and the retention period for the files. If we had elected to backup to tape, this step would be skipped, and the Transaction Log Backup Plan step would follow. For our example database backup, we will adjust the

backup location and retention properties to store only two weeks of backups in the C:\SQL2K Examples\CH4 directory. As we discussed in the previous step, you should refer to Chapter 7 for a complete review of designing and configuring a backup strategy. For now:

27. Select the "Use this directory" option, and specify C:\SQL2K Examples\ CH4 in the location field.
28. Check the "Create a subdirectory for each database" check box to create a new directory for your backup files. The new directory will be the database name, or Southwind, for our example.
29. Check the "Remove files older than" check box, and overwrite 4 with **2** to keep two weeks of backup files and delete any files older than that. In most production backup strategies, you will automatically move the backup files to tape for archiving or back up directly to tape, so this option is not applicable; for simple database plans, though, the convenience of managing backup files is taken care of by SQL Server.
30. Select the Next button to continue creating our maintenance plan.

The backup process will automatically create a unique filename for each backup to disk using the database name and a timestamp value (specified as _YYYYMMDDHHMM) along with the extension you specify. In the case of our Southwind database, an example backup file could be Southwind_200006110401 .BKP and would be located in the C:\SQL2K Examples\CH4\Southwind folder.

Maintenance Plan Wizard: Specify the Transaction Log Backup Plan

Along with backing up our database, we need to back up our transaction log in order to be able to recover our database changes in the event of failure. This step in the Maintenance Plan Wizard allows you to enable transaction log backups and specify the backup options such as checking our backup integrity and which device to use for our backup. We use the same backup options we selected for our full database backup and specify a daily interval so that our transaction logs are backed up each night:

31. Verify that "Back up the transaction log as part of the maintenance plan" is checked.
32. Verify that "Verify the integrity of the backup when complete" is checked.
33. Verify that "Location to save the backup to" is set to Disk.
34. The schedule is already set to a daily backup at 12:00:00 am (midnight), Monday through Saturday. We will not back up the transaction log on Sunday because we already configured a full database backup on Sunday, which will include the transaction log.
35. Select the Next button to continue creating our maintenance plan.

Maintenance Plan Wizard: Specify Transaction Log Backup Disk Directory

As with our regular database backups, you need to specify where the Maintenance Plan Wizard should put your transaction log backups, along with how long the backups should be kept. We will locate the transaction log backup files in the same location as our backup files and keep two weeks' worth of transaction log backups:

36. Select the "Use this directory" option, and specify C:\SQL2K Examples\ CH4 in the location field.
37. Check the "Create a subdirectory for each database" check box to create a new directory for your backup files. The new directory will be the database name, or Southwind, for our example.
38. Check the "Remove files older than" check box, and overwrite 4 with **2** to keep two weeks of backup files and delete any files older than that.
39. Select the Next button to continue creating our maintenance plan.

Maintenance Plan Wizard: Reports to Generate

To monitor the activity of your maintenance plan, you can specify that text file reports should be archived to record activity. You can locate these reports on a shared network drive and review them to verify that your maintenance plans are operating correctly and the health of your database is stable.

In addition to text reports, you can also configure on your SQL Server "operators" who can be notified of activity or errors that occur on your server. *Operators* are individuals who can be notified by e-mail, by pager, or across the network using NET SEND messages. You should configure your server administrators as operators so that they receive messages from SQL Server as events occur. For more information on operators, please refer to Chapter 6 .

For our database maintenance plan, we will record our status reports for the example directory:

40. Check the "Write report to a text file in directory" check box and enter **C:\SQL2K Examples\CH4** in the directory field.
41. Check the "Delete text report files older than" check box, and leave the default retention period of 4 weeks.
42. Select the Next button to continue creating our maintenance plan.

Report filenames are automatically generated based on the maintenance plan name and a timestamp value with the format _YYYYMMDDHHMM. For example, we are going to save our maintenance plan as Southwind Database Maintenance Plan, so an example report file could be named "Southwind Database Maintenance Plan_200012041023.txt" and located in the C:\SQL2K Examples\ CH4 directory.

Maintenance Plan Wizard: Maintenance Plan History

In addition to text file reporting and operators alerts, you can also archive maintenance plan history to the sysdbmaintplan_history system table located in the msdb database on the local server or to a remote server for centralized logging. This ability is particularly useful for administrators who have to monitor multiple servers and want to generate reports from the database. For our example database, we will select the default option of writing maintenance plan history to the local msdb database:

43. Verify that the default option of writing activity to the local server is selected, and click Next to finish creating our maintenance plan.

Maintenance Plan Wizard: Completing Your Maintenance Plan

We have finally reached the end of the Maintenance Plan Wizard. In the "Plan name" field, enter **Southwind Database Maintenance Plan** . You can review all the settings we have specified by scrolling through the summary window. Click the Finish button to complete the wizard and generate your maintenance plan. Click OK on the "successfully created the maintenance plan" dialog box to return to Enterprise Manager.

After you have completed creating your maintenance plan, you can review or modify the settings you selected as well as review its history from Enterprise Manager. To review your maintenance plan:

1. Open Enterprise Manager, and navigate to your server icon.
2. Expand the server icon, and locate the Management folder.
3. Expand the Management folder, and locate the Database Maintenance Plans object. Select the Database Maintenance Plans object to display a list of defined maintenance plans.
4. To review or modify the properties of your maintenance plan, right-click your maintenance plan and select the Properties menu option from the context menu to display the Database Maintenance Plan dialog box shown in Figure 4.12 . From this dialog box, you can modify any of the settings we previously specified using the wizard.

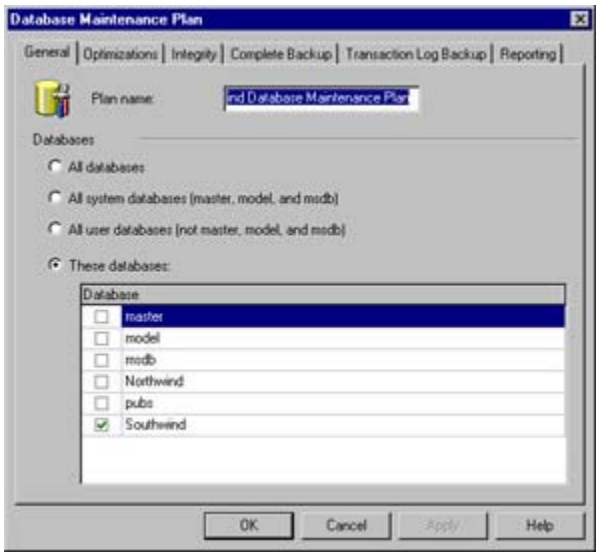


Figure 4.12: The Database Maintenance Plan dialog box.

5. To review the job history of your maintenance plan, right-click your maintenance plan and select the Maintenance Plan History… menu option from the context menu to display the Database Maintenance Plan History dialog box shown in Figure 4.13 . Using this dialog box, you can query the sysdbmaintplan_history system table, where the history information is stored, to determine the success of your maintenance activities.

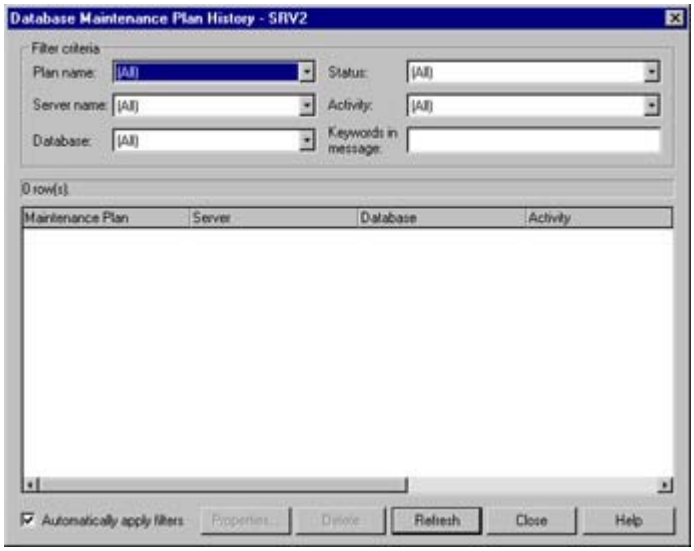


Figure 4.13: The Database Maintenance Plan History dialog box.

Database Modeling Tools

Most of this chapter has focused on developing a physical database plan and creating a new database in SQL Server using the tools that are available. No discussion of creating a database is complete without reviewing the next step in the process: creating a database entity-relationship model. Recall that earlier in this chapter, we defined a logical database model that outlined the data entities (or objects) that will exist in our database. We used this model to determine how to best create our physical database to maximize read/write performance and fault tolerance. In reality, two groups of individuals would be involved in the initial design process: database administrators and database designers. Although it is common for both of these roles to fall on one group or even one individual, the knowledge and activities of each can be distinctly different, and each group requires some level of experience in the other role:

- **Database administrators** work with designers to develop a physical database design and implementation plan as well as continued monitoring and maintenance activities.
- **Database designers** are responsible for defining application requirements that include outlining the database entities, defining relationships between those entities, and modeling the actual database tables, views, and other objects.

Although the concepts and processes involved in entity-relationship diagramming or database modeling are the subject of many books unto themselves, here we briefly review the process and the database designer tool available in SQL Server so that you can begin to explore database modeling and create the actual database objects, such as tables and views, in the physical database we created in the first part of this chapter.

Entity-Relationship Diagrams

Entity-relationship (ER) diagrams are the blueprints for database applications in OLTP systems. The process of creating ER diagrams is well documented and involves:

- Identifying database entities (tables)
- Defining entity attributes (columns)
- Identifying unique row identifiers (keys)
- Defining relationships between entities

The data model then goes through a process called *normalization* that includes three primary rules for efficient data storage. For a detailed description of the normalization process, refer to the “Database Normalization” sidebar in this chapter.

Database Normalization

Designing your database model is dependent on how your database will be used. OLTP systems are designed around a relatively standardized process called normalization. After

you have completed the tasks of entity discovery or identifying the logical data entities in your system, the normalization rules provide guidelines for fine-tuning your data model to optimize performance, maintenance, and querying capabilities. Having said that, complete normalization is not always the best solution for your database. OLAP systems and some OLTP application requirements often result in a denormalized database or at least a denormalized segment. In OLAP solutions that typically contain mass amounts of historical data, the denormalized structure, including multiple copies of data and derived columns, can significantly increase analysis performance and justify its violation of normalization rules. The choice of complete normalization is always dependent on how your database will be used:

Normalization is a process of organizing the tables in a database into efficient, logical structures in order to eliminate redundant data and increase integrity. The physical results of normalizing a database are a greater number of smaller tables that are related to each other. Although there are up to seven normalization rules, called *forms* , the first three forms of normalization are the most significant and commonly used. The remaining normal forms are primarily academic. The primary normal forms are:

- **First normal form (1NF)** Eliminate repeating groups and nonatomic attributes (or fields that contain multiple values).
- **Second normal form (2NF)** Eliminate partial dependencies.
- **Third normal form (3NF)** Eliminate nonkey dependencies and derived columns.

In order for the tables in your database to comply with the 1NF:

- They must have no repeating groups.
- Each field must be atomic (contains no multivalue data).

So what does that mean? The easiest way to understand this concept is to take a look at an example of a table that needs to be normalized. Table 4.5 is an unnormalized table representing cities.

Table 4.5: An Unnormalized City Table

State(Key)	Governor	City(Key)	Founded	Years Old	Founders	Suburb1	Suburb2
NY	Pataki	Roberton	12/28/1941	59	Erwin, Patton	Michaelville	Alexopolis
NY	Pataki	Willville	8/24/1932	68	DeWolf	Auburn	

After you inspect the table for a moment, you will see that Suburb1 and Suburb2 are a repeating group. One of the reasons that repeating groups are troublesome is that they restrict how far you can extend your database to include all related information. After all, what would you do with a third suburb in this case? Another problem is that unnormalized database designs waste space. In this example, cities that have no or one

suburb will not fully utilize the allocated space for the row. Finally, these designs make searching and sorting cities by their suburbs difficult. To eliminate the repeating group, you need to create a new entity called Suburb and form a relationship between the two entities.

There is another problem with the City table: Founders is not atomic, because it contains more than one value that can be split. Again, this design will prevent you being able to sort or search on the data effectively. We could try to resolve this problem by splitting Founders into Founder1 and Founder2. However, this would create a repeating group like the one we had with Suburb1 and Suburb2. Once again, the solution is to add a new entity that represents the founders related to a city. After we add the two entities, we will comply with the 1NF and our tables should look like Tables 4.6–4.8 .

Table 4.6: A Revised City Table

State (Key)	**Governor**	**City (Key)**	**Founded**	**YearsOld**
NY	Pataki	Roberton	12/28/1941	59
NY	Pataki	Willville	8/24/1932	68

Table 4.7: A Founder Table

State (Key)	**City (Key)**	**Founder (Key)**
NY	Roberton	Erwin
NY	Roberton	Patton
NY	Willville	DeWolf

Table 4.8: A Suburb Table

State (Key)	**City (Key)**	**Suburb (Key)**
NY	Roberton	Michaelville
NY	Roberton	Alexopolis
NY	Willville	Auburn

We could not conform to the 2NF until we had complied with the 1NF because. each layer of normalization builds on the previous layers. In order to conform to the 2NF, the tables must follow these guidelines:

- All nonkey fields must be related to all key fields.
- The tables must comply with the rules of the 1NF.

When you look at our current schema, you can see that City is in violation of the 2NF because Governor is dependent only on State, not on City. Once more, the solution is to add a new entity that stores the governor information about the state. One of the benefits of conforming to the 2NF is that you will remove repetitive data, because you have to store the Governor of Ohio only once. This saves storage space and keeps that data

consistent because they are entered and stored only once. Tables 4.9–4.12 show your tables in the 2NF.

Table 4.9: The 2NF City Table

State (Key)	City (Key)	Founded	YearsOld
NY	Roberton	12/28/1941	59
NY	Willville	8/24/1932	68

Table 4.10: The 2NF State Table

State (Key)	Governor
NY	Pataki

Table 4.11: The 2NF Founder Table

State (Key)	City (Key)	Founder (Key)
NY	Roberton	Erwin
NY	Roberton	Patton
NY	Willville	DeWolf

Table 4.12: The 2NF Suburb Table

State (Key)	City (Key)	Suburb (Key)
NY	Roberton	Michaelville
NY	Roberton	Alexopolis
NY	Willville	Auburn

After we have completed normalization through the 2NF, we can further normalize by reviewing the rules of the 3NF. In order to conform to the 3NF, the tables must follow these guidelines:

- Nonkey fields cannot be dependent on any other nonkey field.
- Remove any derived or computed columns.
- The tables must comply with the rules of the 2NF.

If you review our current schema, you'll see one obvious violation of the 3NF: the YearsOld column in our City table. YearsOld is a derived column based on the current date and the Founded column of the City table. Physically storing this information is both a waste of space and a potential maintenance and accuracy problem. To comply with the 3NF, we must remove the YearsOld column. Maintaining derived data like these can introduce inaccuracies, producing application failures. Consider if this field was used to determine a city's eligibility for financial rewards that are based on a city's anniversary. Now our City table in the 3NF will look like the one in Table 4.13 .

Table 4.13: The 3NF City Table

State (Key)	City (Key)	Founded
NY	Roberton	12/28/1941
NY	Willville	8/24/1932

After you have completed normalization through the 3NF, you have designed an efficient database model that will offer optimized storage requirements, data maintenance, and querying capabilities for OLTP systems.

In the following sections, we use the designer tool available in SQL Server to create a simple ER diagram for our Southwind database. From our ER diagram, we can create our database tables to store information for our application. Our ER diagram will be based on the logical model we created in Figure 4.7 .

SQL Server Database Designer

Included in the Enterprise Manager utility is the ability to design ER models using visual database diagrams. You can define all the tables, columns, column properties, relationships, and even indexes using the Database Diagrams editor. Based on our logical database model, we can create the ER diagram depicted in Figure 4.14 . This diagram outlines the detailed structure and relationships among the suppliers, products, orders, and customer logical entities we previously defined for our Southwind database. If we delve into the properties of our tables and relationships, you can easily see how powerful this designer tool is for creating your database ER diagrams and implementing them in your SQL Server database.

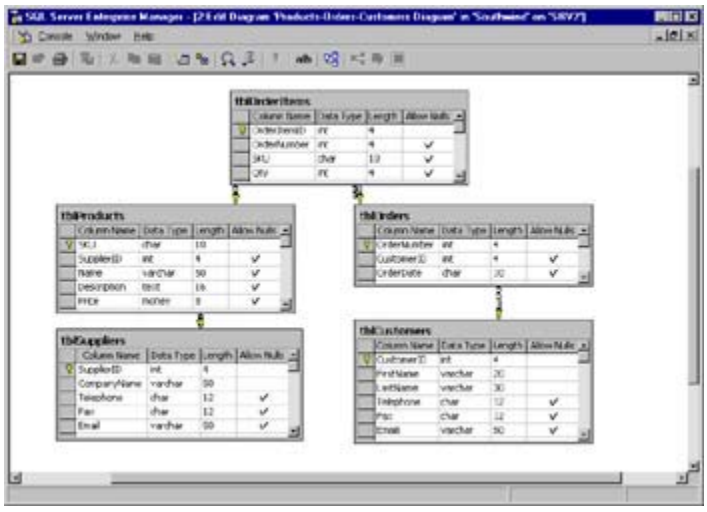


Figure 4.14: Southwind database entity-relationship diagram.

Although the diagram in this model is overly simplified for the purpose of demonstration, the database designer in SQL Server is capable of supporting full database models containing tens or even hundreds of tables. To open the Database Designer tool:

1. Open Enterprise Manager, and navigate to the Southwind database.
2. Right-click the Diagrams container, and select the New Database Diagram menu option from the context menu to display the Create Database Diagram Wizard dialog box.
3. Using the wizard, we could select already existing tables to include in our diagram for relationship modeling. However, because our database is new and we have no tables, you can select the Cancel button to close the wizard and view the new diagram window.

Using the New Table toolbar button or the right-click context menu New Table item, you can add new tables to our diagram. Using the diagram in Figure 4.14 as a guide, try to create the five tables that make up our Southwind database.

After you have defined the Southwind tables, you can change their filegroup placement, create relationships with other tables, and define indexes using the Table Properties dialog box shown in Figure 4.15 . To open the Properties dialog box, right-click the table in your diagram, and select Properties from the context menu. If you review the Properties dialog in Figure 4.15 for the Orders entity in our database, you can see that we can easily specify in which filegroup to store the data to meet our original physical database plan.

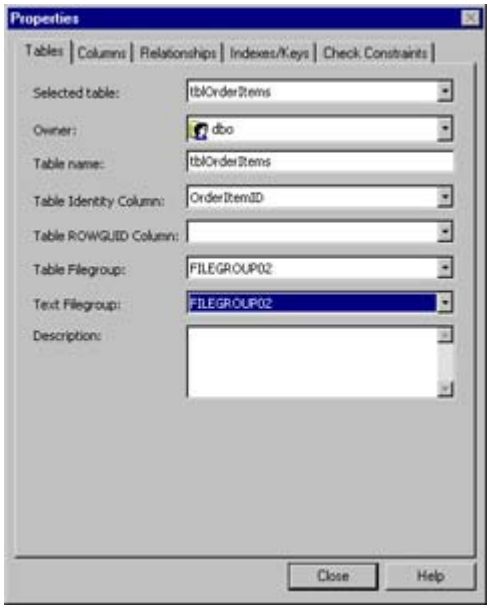


Figure 4.15: The Table Properties dialog box.

After you have made changes to your database diagram, you can save the changes directly to your database, or you can save your modifications to a T-SQL script, which can be executed on any server or backed up for future use. To save your new database

diagram and create the Southwind database tables, click the Save button on the designer toolbar. After you have saved your database tables, you can continue to edit them or begin working with your database application.

Although there are numerous popular database modeling tools and every designer has a favorite, you should consider taking a close look at the Database Designer diagrams tool in SQL Server Enterprise Manager. Its use can be very efficient for smaller database design projects, and best of all, it's included with SQL Server, so it's readily available for use.

Summary

SQL Server applications begin with databases. As a fundamental component in your data-dependent applications, understanding how information is stored in SQL Server and the most efficient layout for your database is critical to its performance. SQL Server 2000 has two types of databases: system and user. System databases contain the configuration information for your server, your databases, and features such as SQL Server Agent and replication. Every installation of SQL Server includes the Master, msdb, Model, and TempDB system databases. User databases are the databases that are created for your applications to store and manage data.

Databases in SQL Server are made up of data and log files, which are responsible for storing the data and transaction logs for your database. Transaction logs are records of changes to the data in your database. Transaction logs are useful for recovery because they record each change to the data. Data files are used to store the actual data contents of your database. Data files are physically made up of 8K pages. With the exception of text and image data, the 8K-page size represents the maximum single record size. Pages of data file space are allocated to tables and indexes in groups of eight, which is called an *extent* . In case a table or index does not require 64K (8 x 8K) of storage, SQL Server supports mixed extents, which allow for a single extent to be allocated to a maximum of eight different tables or indexes for more efficient use of data file space. Information in transaction logs is stored sequentially as log records.

SQL Server databases can be made up of multiple data and log files. Using multiple files, you can support increased growth requirements in addition to segmenting your database files across multiple physical disk systems. Data files placed on separated disks in SQL Server can be combined into a logical unit called a filegroup. Tables, indexes, text, and image data can be allocated to different filegroups. By dispersing your data across multiple disks, you can optimize system performance. Each data or log file in SQL Server can be enabled for autogrow, which allows data or log files to automatically expand to provide additional space to meet increasing growth requirements.

Designing your physical database structure begins with identifying your logical database model. Your logical database model consists of identifying the data entities in your database application. Examples of data entities in an order-entry system could be customers, orders, products, and suppliers. After you have identified your data entities, you need to determine entity read/write activity. By identifying the amount of read and write activity that your entities will undergo, you can efficiently organize your database by allocating the entities to different filegroups. Although many smaller database solutions operate efficiently with a single database file, larger systems can take advantage of multiple data files to partition data across numerous physical disks.

The process of designing the physical layout of your database involves selecting the appropriate RAID level for your disk configurations. RAID offers both increased performance and fault tolerance. Increased performance is provided through striping, the process of spreading the data across multiple physical disks. Fault tolerance in RAID

systems is delivered through mirroring or parity data. Mirroring stores two copies of the same data on different physical disks. Parity is the process of calculating a parity bit that can be used to rebuild data on damage disks. This parity information is spread across multiple disks so that the contents of any single disk can be rebuilt if the disk fails.

After you have structured your physical database based on your logical database model, you can begin designing your database application. Designing your database application involves creating an entity-relationship diagram and undergoing a process called normalization. ER diagrams lay out the entities in detail, defining their columns, or attributes, column properties, and relationships with other tables. The process of normalization outlines three common rules that work to reduce redundancy, reduce dependency, and deliver a more accurate and manageable data structure.

FAQs

Q. Can I store my database files on a compressed drive to save space?

A. No. Microsoft recommends that you do not put any of the files on a compressed drive. The compression and decompression process will impact the performance of SQL Server and can cause database integrity problems.

Q. Do I have to use Enterprise Manager to create my databases, tables, and other objects in SQL Server 2000?

A. No. Enterprise Manager provides graphical design tools and wizards, but you can also use T-SQL to write scripts to create all your database objects in SQL Server. Many administrators prefer using T-SQL scripts because they can be backed up and reused.

Q. What are all the databases that are installed with SQL Server 2000? Can I delete them all?

A. No. SQL Server includes four system databases that cannot be deleted or your database server's stability will be impacted. These system databases are msdb, Model, TempDB, and Master. The remaining two databases that are installed with SQL Server are the Pubs and Northwind databases. These are sample databases that can be deleted without affecting SQL Server.

Q. Where should I place my transaction log file? Can I use more than one?

A. Transaction log activity is sequential and primarily writes. For this reason, you should isolate your transaction log on a separate physical disk to minimize read/write head movement and eliminate contention over data file activity. Additionally, you can place your transaction log on mirrored disks for fault tolerance. You can use more than one transaction log file if you need additional storage, but transaction logs are filled in order, so you cannot use multiple log files for performance gain.

Q. Do I need to use multiple data files and different disk systems for my database application?

A. Not necessarily. Microsoft states that most databases will receive sufficient performance using a single data file and single transaction log file. Large applications or databases with multiple "hot spots," or areas of high read/write activity, can be optimized using multiple disk systems and data file placement to reduce the amount of work required of single disks.

Chapter 5: Database and Server Security

Introduction

Although data are always considered a priceless asset in most organizations, and protecting data is a high priority, new applications such as electronic commerce, or e-commerce, have opened additional pathways to the vault through public network connections such as the Internet. Even though most organizations remain unscathed by attacks, it is not without significant effort spent planning and implementing security in the organization. By combining server- and file-level security provided by Windows 2000 with SQL Server 2000's user- and role-level security models and encrypted network communications, you can protect your data, the heart of your organization. SQL Server 2000 offers configurable user and role security levels for database and object access. With features such as replication and database access over the Internet, multiprotocol encryption and SQL Server's new support for SSL can protect your data on the wire. SQL Server 2000's integration with Windows 2000 Active Directory adds support for Kerberos, context delegation, and centralized management.

This chapter reviews the security architecture of SQL Server 2000 and the available security modes. As you work through the chapter, you will create users and database roles and assign permissions to the database and its objects, such as tables and views. To further secure your server, you will learn the advantages and disadvantages of multiprotocol encryption and SSL support in SQL Server.

Planning SQL Server Security

SQL Server Security, or any system's security for that matter, is easiest to manage when you start with a plan. You might be tempted to start with a rather relaxed view toward your security configuration if you have only a few databases, objects, or users. Don't give in to that temptation. As systems grow, if their security model doesn't support that growth, security holes will appear. In addition to planning for the future, a good security model can make your administration easier today. A well-planned security model is much simpler to administer than one that is not logical, well documented, and standardized.

To properly secure a SQL Server 2000 installation, you need to take a number of factors into consideration. You can't be concerned merely with which users have access to the SQL Server, its databases, and the objects within them. You must also consider the security of the OS on which SQL Server is installed, the file system that contains the data files, and the network that transmits data to and from the clients. Figure 5.1 demonstrates the various levels in which you can deal with security. You might find that your configuration requires security administration at one or more of the layers in the diagram.

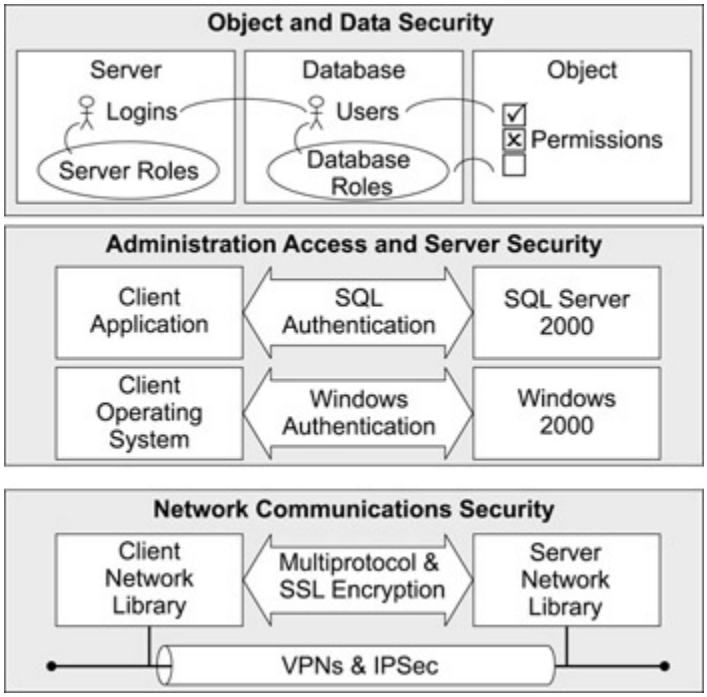


Figure 5.1: The various levels of security in a SQL Server 2000 environment.

As you begin to plan your security model, ask yourself these questions:

- On which operating system will SQL Server be installed?
- Which operating systems will clients use when connecting to SQL Server?
- Which network protocols will be used for communication with SQL Server?
- Do all the clients connecting to SQL Server have an account in a Windows NT or Active Directory domain?

- Will you provide access to your SQL Server's data via the Internet or the Web?
- Will all the traffic between the clients and the server be on a private network, or will data be traveling across public networks such as the Internet?
- Do all users require the same level of permissions to the data, or are there a variety of permissions requirements?

The answers you get to these questions will provide an excellent foundation for the information required as you begin designing your security model.

Understanding SQL Server Security

SQL Server 2000 security depends a great deal on the operating system on which it is installed. When running SQL Server 2000 Personal Edition on Windows 98, for example, you cannot use Windows Authentication mode; only SQL authentication can be used. In addition, since Windows 98 does not have file-level security, the files that make up the SQL databases remain unprotected in the file system. When SQL Server is installed on a Windows 2000 system, however, it can rely on the Windows security and file subsystems to authenticate users and secure the SQL database files on the hard disk.

Combining the strengths of a secure network infrastructure, a properly configured OS and Windows domain model, and the internal object-level permission capabilities of SQL Server, you can create a very secure SQL Server environment that is surprisingly simple to administer.

Manage By Groups

Windows NT has always used, and now Windows 2000 uses, a group-based administration model. You might have read about the Windows AGLP method for managing users and groups. AGLP stands for accounts (A) go into global (G) groups, which get placed into local (L) groups that have permissions (P) assigned to them. Use global groups to organize users with different needs at the domain level. Use local groups at the computer on which the resource is to assign permissions to resources. Place the global groups of users into the local groups that give them permissions to the resources they need, and voilà—the users in the global groups now have access to the resource.

In SQL Server, you can, of course, still use the AGLP method—just consider SQL Server the resource, or you can modify the AGLP method to be an AGDRP solution and achieve the same benefits. An AGDRP solution is one in which your Windows accounts get placed into global groups. Those global groups are granted login access to SQL Server and given a user account in the database. The user accounts are placed into database roles (DR) in the database, and the permissions are assigned to the database role.

The beauty of this model is that as new users join the organization or as existing users leave, you merely add or remove their Windows account in the global group at the domain level, and you are done. If you need to modify the permissions, you modify them

only once—for the database role to which they are assigned—and all users immediately benefit from the permissions change.

C2 Certification

SQL Server 2000 has achieved a C2 rating in accordance with the National Security Administration (NSA) Trusted Computer System Evaluation Criteria (TCSEC) specification. What does C2 certification mean? A C2 classification indicates that the system offers user identification and access control mechanisms, the ability to audit user actions as they relate to security, and the ability to restrict or grant access to resources based on a user's identity. Certain military, government, and corporate facilities demand a particular minimum level of security capabilities in their systems. The NSA's TCSEC classifications provide a set of rules to measure systems by evaluating their security strengths and weaknesses.

SQL Server 2000's C2 configuration is comprises SQL Server 2000, with no SQL service packs, running on Windows NT 4.0 Service Pack 6a, with the C2 update hot fix applied. If you are interested in configuring your system to be C2 compliant, you can get more information from Microsoft at www.microsoft.com/technet/security/sqlc2.asp.

An important part of the C2 implementation of SQL Server 2000 is auditing. *Auditing* is the process of recording users' actions. The system can record, in an audit log for review by the administrator, actions such as a successful or failed login attempt or object access. SQL Server has traditionally had relatively basic auditing capabilities. With SQL Server 2000, those capabilities have been extended into the SQL Server Profiler utility. The Security Audit event category was added to Profiler to provide a more complete set of audit events. You can use the Profiler tool to create complete, customized audit traces to run against your servers. Microsoft, however, has provided a server configuration option to allow administrators to conveniently configure a basic C2-compliant audit on their servers. To enable this option, use the sp_configure stored procedure. The "c2 audit mode" option is an advanced option and as such is available to you only when advanced options are shown. To turn on C2 auditing on your SQL Server 2000 system, execute the following code, then restart your SQL server:

```
USE master
EXEC sp_configure 'show advanced option', '1'
RECONFIGURE
EXEC sp_configure 'c2 audit mode',1
RECONFIGURE
```

Once auditing has been turned on, the server will begin writing audit information into trace files in the \mssql\data directory (or the \mssql$Instance\data directory for a named instance of SQL Server 2000). A new audit file is created each time SQL Server is restarted or when the current file reaches 200MB in size. If there is no remaining free

space on the drive on which audit files are stored, auditing cannot continue. When auditing stops, the SQL Server instance is shut down immediately, as a security precaution. You can free some space on the drive and restart SQL Server, or you can restart without auditing SQL Server using the –f parameter.

Warning When an audit trace is running on a system, it could be a significant impact on the system's performance. Rather than incurring that performance hit on your server, you can configure SQL Server Profiler to run on another machine. This requires that you create your own audit trace definition and run it in Profiler. The "c2 audit mode" option always runs on the server that is being audited.

In addition to the performance overhead, active systems can generate a large amount of auditing data. As these audit data are saved to disk, the drive that stores the audit trace files can fill up rather quickly. If you are using the "c2 audit mode" option, SQL Server will shut down if no audit data can be written. You need to keep an eye on the free space of the drive that stores the audit trace files, and free additional space as the drive nears capacity.

Administration Access and Server Security

The subjects of administration access and server security actually cover a number of topics. When dealing with these issues, you need to consider who has access to view and modify SQL Server's configuration, who has access to the SQL Server files on the drive, and who has access to connect to SQL Server to use its services and access its data.

Configuring the SQL Server 2000 Service Accounts

SQL Server 2000 runs as a service on Windows 2000. A *service* , for the purposes of this discussion, is a program that runs in the background on a system, regardless of who might or might not be working on that computer. Services run in their own memory space, and they run as their own security identities.

When you install SQL Server, you have the choice of having your SQL Server service (MSSQLServer) and the SQL Server Agent service (SQLServerAgent) run as either the local system account or as a specific user account that has been created for them as either local Windows users or Windows NT or Active Directory Domain users.

The local system account is an account that exists on every Windows NT or Windows 2000 system. This special account has full access to the local computer but has absolutely no access to network resources. This can be a problem if your SQL Server service needs to access remote network resources. For example, if you wanted to have your SQL Server back up to a share on another computer on the network, it would be unable to do so if it were running as the local system account. If you wanted to implement the SQL Mail feature to access mail in an Exchange mailbox on another server, again, you would be unable to perform this task. The local system account has no network access abilities.

If you need the SQL Server service to have access to remote resources, it must run as a user account that can be recognized by those remote systems. The preferred way to do this is to create an account in a Windows NT or Active Directory domain that can be referenced by all the computers involved. For the rest of this chapter, the user account created for the service will be referred to as the *service account* .

The service account also needs to be given the appropriate permissions on the various systems it will access. For the SQL Server itself, generally, the service account should have administrative access. In other words, the service account should be made a member of the built-in administrators group on the system on which SQL Server will be running. This is because the SQL Server service needs to be able to create files, manipulate files, delete files, access system resources, and read and modify the registry.

If you are uncomfortable having the service run as a local administrator, you can make the service account a normal user and assign the additional permissions required to let it do its job. The C2 configuration of SQL Server 2000 actually requires that the service account *not* be a member of the local administrators group. If you want to have the service account run as a nonadministrative user, you need to configure its file and registry permissions and user rights manually. The following is a list of rights and permissions as defined by the *C2 Administrator's and Users' Security Guide* . You can get the guide from Microsoft's Web site at www.microsoft.com/technet/security/sqlc2.asp. The rights needed by the service are:

- Act as part of the operating system
- Increase quotas
- Replace a process level token
- Log on as a service

The file permissions required are:

- Full control of the \MSSQL subdirectory
- Full control of all .MDF, .NDF, and .LDF files

The registry permissions required are:

- HKEY_LOCAL_MACHINE\Software\Microsoft\MSSQLServer
- HKEY_LOCAL_MACHINE\System\CurrentControlset\Services\ MSSQLServer

  (or ….\MSSQLServer$InstanceName for installed instances)

- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\ CurrentVersion\Perflib

Remember that you need to assign these rights and permissions only if you do not have the services running as either the local system account or as an account that has local administrator privileges.

You can confirm or change the service account on which your SQL Server 2000 instance runs from a number of tools. Enterprise Manager, the management tool that comes with SQL Server 2000, is a convenient way to do this. Open Enterprise Manager, then find and highlight your SQL Server 2000 in the tree. Right-click the server, and select Properties from the menu. The Properties window will open. Switch to the Security tab. Notice that the bottom section of the tab contains settings for the "startup service account. Figure 5.2 shows the security settings for a SQL Server named SERVER2. You can see that the service is starting not as the local system account, but as a specific account named MYCORP\SQLService. MYCORP\SQLService is the Windows domain notation to identify the SQLService account that exists in the MYCORP domain. The correct password for the account needs to be entered in the Passwordfield.



Figure 5.2: SQL Server 2000 service startup service account settings.

Securing the SQL Server 2000 Executable and Data Files

Users who connect with SQL server do so using a SQL server client application. It is possible that the Windows operating system on which the SQL server is running will authenticate the user's identity, but the user does not actually need any permissions to the computer's file system or local resources. This is because all access to data in a SQL service is provided by the SQL Server service itself. In the previous section , we discussed creating a service account for the SQL Server service and ensuring that the service account had full access to the data files. As long as the SQL Server service has access to the files on the server, it will access those files, on behalf of its users, to read and store data. All access to data and objects is managed using SQL Server's internal security mechanisms. Figure 5.3 demonstrates the relationship between a SQL client, the SQL Server, and the executable and data files used by the SQL Server service.

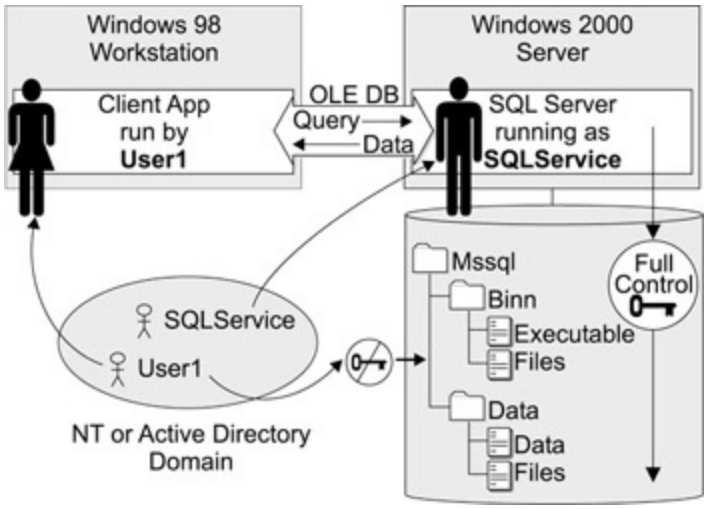


Figure 5.3: SQL Server file system security.

Object and Data Security

SQL Server 2000 offers a very flexible security model to you, the administrator. You have the ability to control access to the server itself, give a single login access to one or more databases, and, within databases, restrict users' interaction with the objects in a database (tables, view, stored procedures, and the like) while giving other users full access to the objects. In addition, you have the ability to integrate the Windows authentication mechanism to validate users as they connect to SQL Server. Figure 5.4 provides a very generic diagram of how the Active Directory Domain, the local Windows system, SQL Server itself, its databases, and object-level permissions all interact with each other.

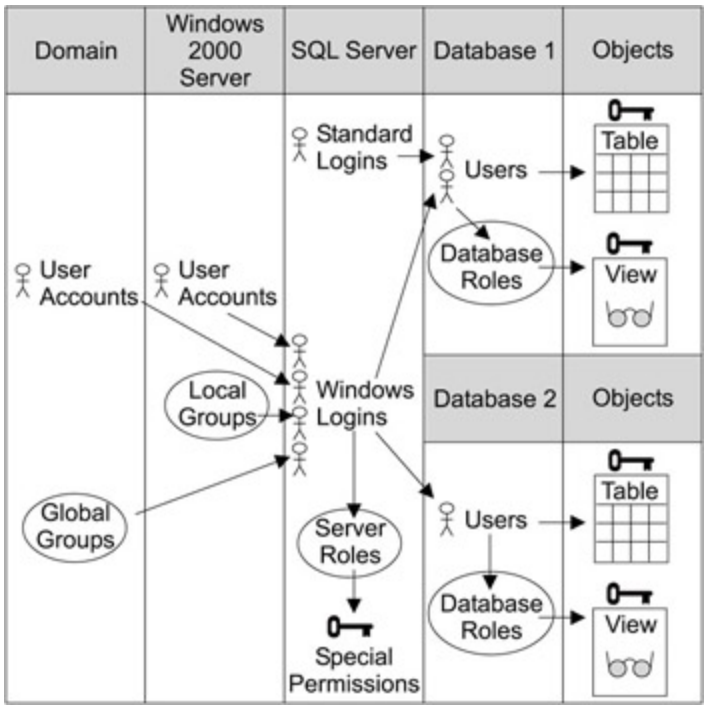


Figure 5.4: An overview of the SQL Server security model.

Figure 5.4 contains a number of columns. The column farthest to the left represents the Windows NT or Active Directory domain. The second column represents the users and groups that are created on the Windows system on which SQL Server is installed. The middle column represents the first level of access to SQL Server. It is at this level that we find Logins to SQL Server and Fixed Server Roles. The fourth column represents the databases and contains user accounts and database roles. Finally, the fifth column represents access to the objects in a database and the permissions assigned to various users and database roles.

Securing Login Access to SQL Server

In order for users to get to the data provided by SQL Server, they must first log in to SQL Server. This is the first layer of security that is implemented inside SQL Server itself. Prior to this point, only domain and local system security settings were being dealt with.

SQL Server logins control which individuals or groups of individuals have permissions to connect to a SQL Server. Logins do not specify which databases are available to those individuals, or what can be done with the objects in the server's databases. SQL Server logins merely control access to the SQL Server itself.

If you review Figure 5.3 , you will notice that there are two basic types of logins. There are logins that are created internally in SQL Server and logins that reference existing Windows users or groups. The *standard logins* are created by the SQL administrator internally in SQL Server and are primarily for non-Windows or remote users to use to log in. SQL Server can also grant login access to existing Windows users or groups. This is,

in fact, the default type of login and the preferred way to allow users to gain entry to SQL Server. The methods to create logins and the differences between Windows logins and standard logins are discussed later in the chapter. For now, you need to understand only that a login of one kind or the other is required for a user to gain access to SQL Server.

Assigning Privileges to Perform Serverwide Operations

At times, as the administrator, you need to allow other people to perform certain administrative activities on the SQL Server. By default, individuals with login access to SQL Server have no implied privileges on the server. If you require specific logins to be allowed to perform administrative tasks on the server, you need to give them the permissions to do so.

Server roles are basically groups that exist at the server level. The server roles are built into SQL Server and have specific permissions preassigned to them. Many of these permissions cannot be acquired any other way. By adding logins to these Server Roles, you allow those logins to perform the actions for which the role has permissions. For example, adding a login to the dbcreator server role allows those users to create databases in SQL Server. The set of roles that exist at the server level and the permissions assigned to them are all predefined. You cannot create your own server roles, nor can you change their permissions. Because of these restrictions, the server roles are called *fixed server roles* . We discuss the set of server roles and their permissions later in the chapter.

Granting Access to Databases

Logins to SQL Server do not specify the databases to which an individual has access. Rather, logins merely let the individual connect to SQL Server. For an individual to be able to connect to a database, he or she needs to have a user account created for login in the database to which the user wants access. When creating a user account in a database, you specify the login that maps to that user account. If you want all logins to have access to a given database, you can create a “guest” user account in the database. This allows any person with login access to SQL Server to gain entry to the database. You can see the use of the guest account in the Master, MSDB, TempDB, Pubs, and Northwind databases.

There are a couple of things to consider with user accounts. First, user accounts map to logins. A login can be one of many things. A single login could be a standard SQL login, a Windows user account that has been granted login access, or a Windows group account that has been granted access. If the login is a standard login or a Windows user account, there is a one-to-one relationship between the number of people represented by a given login and thus the number of people who access a database as a given user account. If the SQL login is a Windows group account that has been granted login access, however, the situation is different. The Windows group account represents any number of Windows users. The login that grants the Windows group access to SQL Server then maps to a single user account in the database. The end result is that a large number of Windows users are represented by a single user account in the database. That works great because you have to set up only one user account and one set of permissions and rights.

Grouping Users into Database Roles

In any security system, it is generally easier, and therefore recommended, to assign permissions to groups of users rather than to individual users. By creating groups, placing users into the groups, and setting the permissions to the group only, you can significantly cut down on the total number of permissions that need to be assigned and maintained. As new users are added to the system, getting them the permissions that they need is as easy as adding them to the existing group. If permissions need to be changed, you change them once in the group, and all the users in that group are affected. This, of course, remains true inside SQL Server databases. In SQL Server, however, groups are called *roles* .

Earlier we discussed fixed server roles. Server roles exist at the server level and allow their members to perform serverwide operations. SQL Server also provides a number of built-in roles for each database it contains. The built-in database roles, better known as *fixed database roles,* provide a convenient mechanism for assigning special database permissions to users. Some fixed database roles allow granting of permissions that are not available in any other place. The db_securityadmin fixed database role, for example, allows its users to manage permissions, roles, and role memberships in the database. Other database roles offer an easy way to grant blanket permissions to all objects in the database. The db_datareader role allows its members to select from any table or view or rowset function in the database.

Unlike server roles, for which you could not create your own, SQL Server does allow you to create your own database roles. When creating your own roles, you have an option of two types. *Standard roles* have their memberships defined by you when you create the role or on an ongoing basis. You manually add or remove members from standard roles. Once a user has been added to a standard role, that user stays in that role until you remove him or her from it.

Application roles are different from standard roles in that they have no continuous membership. A user is placed into an application role by the client application itself. The developer of the client application works with the DBA to learn about the name of the application role and to be given the special password needed to place users into the role. When the user connects to the database using the application, the application activates the application role behind the scenes. Once the application role is activated, the user gains all permissions that the administrator assigned to the role. The user stays in the role for the life of that connection. As soon as the client closes the application and disconnects from the server, he or she is no longer in the application role. The end result is that the user has the permission needed to run the application, but if the user attempts to access the server with any other tool, the application role is not active and the user will not have the necessary permissions.

Using Views, Stored Procedures, and User-Defined Functions to Simplify Security

Views, stored procedures, and user-defined functions are useful objects to developers. They offer ways to simplify queries, bundle up logic for reuse, and extend the capabilities of SQL Server. These same objects, however, can also be a big help to administrators. Views, stored procedures, and user-defined functions can be tools that you can use to simplify administration of objects in your database. Views can pull data from a number of tables. Stored procedures and user-defined functions can perform actions on a number of tables. Rather than having to assign permissions to all the underlying tables, SQL Server allows you to assign permissions to only the views and stored procedures the user needs. As long as the owner of the underlying tables is the same as the owner of the view, procedure, or function, SQL will honor the permissions and let the user access the underlying tables. If the user bypasses the view, procedure, or function to directly access the base tables, the user will not have permissions and will not be able to gain access to the data.

Network Communications Security

Rarely is a SQL Server solution completely contained in a single machine. As a client/server database solution, SQL Server has one strength in the ability to support numerous users across a computer network. This networked environment poses its own set of security risks, however. Unless specified otherwise, the statements and data that travel between the client and the server are in clear text. Assume that an individual has access to a network segment that carries SQL Server traffic. If that person were to install a packet analyzer (also known as a *sniffer*) on that segment and capture the data as they pass by, the user would be able to read the data without any further decryption required.

You might say that that scenario sounds rather specific, and if you are running your client/server applications over private LANs and WANs, you might be right. Consider this, though: If you are running your applications across a public network, such as the Internet, you have *no* control over which network segments your data travel as they route from client to server and back. On your private networks, the ability to place packet analyzers on network segments can be controlled. On the Internet, however, you have no ability to control access to those segments that don’t belong to you. In these situations, it is more secure to encrypt the data as they leave the client or server and decrypt them when they reach their destination.

A number of options are available when it comes to encrypting data. The network clients, the routers, or the firewalls themselves could provide the encryption. Microsoft Windows 2000 provides a number of encryption solutions, including virtual private networks (VPNs) and Internet Protocol Security (IPSec). We can also utilize the capabilities of the SQL Server network libraries to encrypt our data. All network libraries in SQL Server 2000 can have their data encrypted using Secure Sockets Layer (SSL) encryption. Alternatively, the multiprotocol network library supports encryption for backward compatibility with existing clients.

Security Options in SQL Server

SQL Server 2000's security model is a combination of Windows authentication and internal SQL authentication mechanisms. As the DBA, you have the option of supporting both the Windows and the SQL authentication mechanisms or only the Windows authentication mechanism. Regardless of what type of authentication you use, you can be certain that each user must authenticate one way or another in order to use SQL Server.

Before we go any further, let's define what we mean by *authentication* . In the sense in which we use the term, *authentication* means to verify a user's identity by confirming that the username and password (or credentials) he or she provides during login are correct. Consider this generic example. A username and password have previously been created and placed in some security information store. When connecting to the system, a user is prompted for their username and password. After the user supplies her credentials (her username and password) to the system, the system locates the username in the security information store, confirms that the password is correct, and lets the user connect. If the credentials supplied do not match the data in the security information store, an error is reported to the user and possibly logged in an audit log somewhere.

Understanding the Windows Authentication Mode

Windows NT, and now Windows 2000 with Active Directory, has a very robust security information store. Windows NT or Active Directory domains have advanced security features, including password expiration, length, and reuse restrictions. You might have noticed that you cannot log in to a Windows NT or Windows 2000 system without providing a valid username and password. The network administrator builds the NT domain or Active Directory domain and places the users' usernames and passwords into the domain. As users log in to the system, they supply their credentials. Windows authenticates the users by validating their information against the data in the NT or Active Directory domain and, if the information is validated, allows the users to connect.

Wouldn't it be great if you could simply use that existing Windows authentication mechanism to authenticate users when they connect to SQL Server? That would save your from having to re-enter all those usernames and passwords into a separate data store. It would allow users to connect to SQL Server without being prompted for another set of credentials. It would mean that a user has only one user account—his or her Windows domain user account. There wouldn't be two copies of a user account that need to have passwords synchronized. If a user leaves the company and his or her account is deleted, it limits access to all resources. That would be nice, and it just so happens that you *can* use the Windows authentication mechanism to validate users who connect to SQL Server. In fact, the Windows authentication mode is the default authentication mode for SQL Server.

A final but nonetheless significant benefit to using Windows authentication is network security. The Windows authentication mechanism does not transmit passwords across the

network. Windows instead uses a challenge-and-response mechanism to verify a user's security. A challenge-and-response sequence looks like this:

1. The client connects to the server and requests a challenge phrase.
2. The server generates a random string of characters (called the *challenge phrase*) and returns it to the client.
3. The client then encrypts the challenge phrase using the user's password as the encryption key.
4. The encrypted challenge phrase (not the password itself) is then transmitted back to the server.
5. The server decrypts the response using the user's password, which it got from Active Directory.
6. If the decrypted challenge phrase matches the original challenge phrase, that must mean that the user encrypted it with the same password with which the system decrypted it. Therefore it is a valid password, and the user can connect.
7. If the decrypted challenge phrase does not match the original challenge phrase, the connection is denied, and the user must reauthenticate to connect.

This challenge-and-response mechanism ensures that when Windows authentication is used, no passwords are directly transmitted across the network. This feature offers an extra level of protection when clients are connecting to resources such as SQL Server across vulnerable networks.

Figure 5.5 shows how SQL Server accepts login requests from users with Windows credentials and passes those credentials to Windows for authentication purposes. Windows validates the user and returns to the SQL Server a list that contains the security ID (SID) for the user and the IDs of all the global and local Windows groups to which the user belongs. Once SQL knows the user is who the user says he or she is, it attempts to locate in its list of granted logins the user or one of the groups to which the user belongs. SQL Server stores the list of granted logins in a table called sysxlogins in the master database. If SQL Server does not find the user or any of the groups to which the user belongs as having been granted login access, the SQL Server returns an error and audits the failed login, if auditing is enabled.

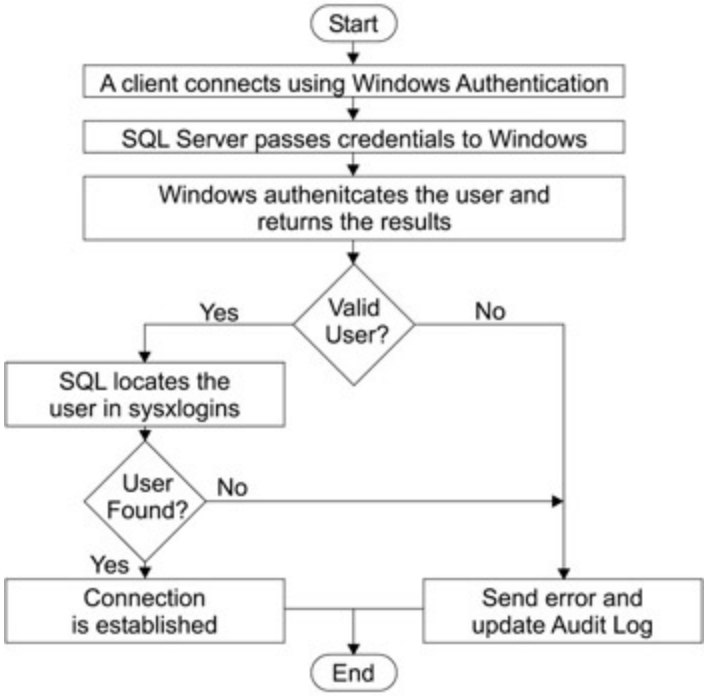


Figure 5.5: The Windows authentication login process.

Granting login access to existing Windows users can be done one of two ways. You could explicitly grant access to each Windows user that needs to log in to SQL Server, or you can grant access to a group of users who all need access to SQL Server. The advantage of using Windows groups over individual user accounts is that you have to grant only one login to SQL Server. On the domain side, as the domain administrator places domain users into the group, those users are automatically able to log in to SQL Server.

Understanding the SQL Authentication Mode

A number of users, however, work on systems that are not part of a Windows NT or Active Directory domain. These users might be running Unix systems or Apple computers or might be attempting to access SQL Server's data from a public Web site, their cell phones, or PDAs. For users who cannot provide valid NT credentials with which to authenticate, SQL server has its own internal authentication mechanism. When SQL Server acts as the authentication mechanism, it stores the usernames and passwords for standard SQL accounts internally. As users connect, they need to supply their credentials. SQL Server validates those credentials against the usernames and passwords it has stored in its own system tables, and then it grants or denies access accordingly. The Windows authentication mechanism plays no part in the process.

In Figure 5.6 , you can follow the process of a standard SQL login connecting to the SQL Server. A client connects using a standard SQL account. When the client does so, he or she must supply a username and password to SQL Server. SQL Server attempts to locate the username in its list of standard SQL logins. This list is kept in the sysxlogins table in

the master database. You might recall from the previous section that Windows logins are also kept in sysxlogins. The difference is that with Windows logins, only a reference to their Windows accounts is kept in sysxlogins. With SQL logins, the entire logins, including their passwords, are stored in sysxlogins. SQL Server uses this information to authenticate the user. If the user is found and has supplied valid credentials, the connection is established and the user is let in. If, however, the user is not found or the user's password is incorrect, an error is returned to the client and logged in the audit log, if auditing is enabled.

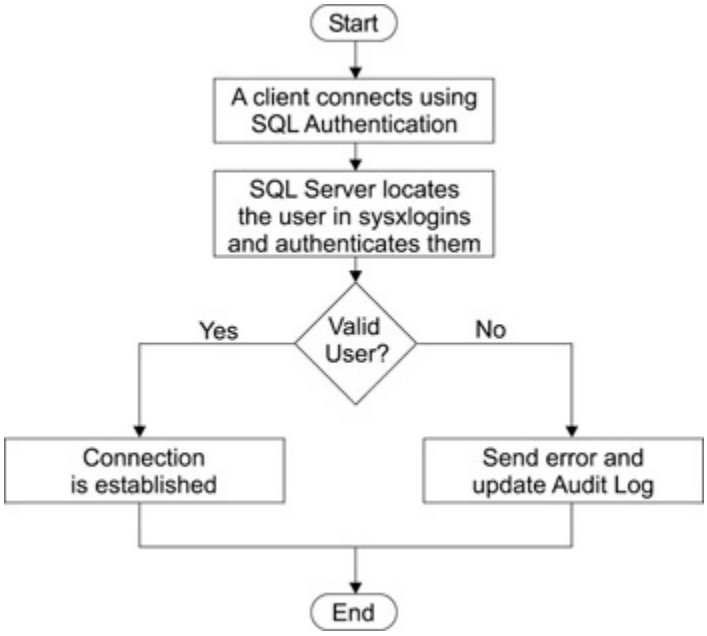


Figure 5.6: SQL authentication login process.

In SQL authentication, usernames and passwords are transmitted across the network as the client establishes the connection to the server. This process is unlike Windows authentication, in which the password itself never crosses the wire. The transmission of SQL passwords across the network might be an area of concern for you. The latest SQL Server ODBC and OLEDB drivers do encrypt the password as it is transmitted across the network, but it is an encrypted version of the password itself. This is less secure than the Windows authentication mechanism, in which the challenge phrase, not the password itself, crosses the network.

Database Users, Roles, and Permissions

SQL Server 2000 is capable of storing up to 32,767 databases per server instance. Take, for example, a server that has both the accounting application's database as well as the contact management application's database. If you consider the users who need access to these databases, you might come up with three or more different groups of users: the accounting users who need access only to the accounting database, the sales users who need access to only the contact management database, and some supervisors and

managers who need access to both databases. SQL Server allows you to grant these different users access to different databases.

Take the previous example one step further. Focus on just the contact management database. Say, for example, that the sales department has two assistants who are responsible for running reports on the contact data. The assistants need permissions to access the data for reporting purposes only. They do not need, nor should they be given, permissions to modify the data. The sales representatives, however, need the ability to modify a given customer's information, add new customers, delete old customers, and so on. Again, SQL server allows you to offer different permissions to various users.

To summarize, SQL Server offers three levels of security:

- Login authentication to gain access to SQL Server
- User accounts in databases to provide access to a specific database
- Permissions to objects (tables, views, and the like) in databases and to perform certain actions

Selecting a Security Mode

In the previous section , Windows authentication and SQL Server authentication were both explained. It might be seen from the preceding section that the Windows authentication mechanism offers a higher level of security than the SQL Server authentication mechanism. A number points can be made in support of this idea. Windows user accounts can be restricted in a variety of ways. The length, complexity, expiration date, and reuse of passwords can be controlled. Windows users can be allowed to authenticate only during predefined time intervals, the stations they can connect from can be restricted, and when the system is used in conjunction with IPSec, the network administrator can even control which protocols, networks, and remote services to which users have access. Add to that the robust nature of Active Directory, and you have a true enterprise-level user and resource management solution. A user account that exists anywhere in the Active Directory structure can be granted access to not only one but *any* SQL Server in the enterprise. If that user changes his or her password in Active Directory, the user doesn't need to also change it in SQL too, because SQL doesn't store the user's password. For a large, multiuser, secure environment, Windows authentication should be the clear choice.

SQL Server authentication, on the other hand, offers a relatively basic security model. Users are given logins and passwords. The logins and passwords need to be created and stored on each SQL Server to which the user needs access. If the user changes a password on one server, it has no effect on the user's passwords on other servers. In other words, users must separately maintain the password for every SQL Server login they have. There are no restrictions on the length, complexity, expiration, or reuse of passwords. Neither can you control how, when, or from where a user connects. SQL Server authentication should be used only when Windows authentication cannot be used—for instance, because you need to offer access to a user who does not have a Windows account or because users

are connecting from Windows systems that are not recognized by your NT or Active Directory domain.

You should keep the preceding information in mind as you select the security mode to support on your SQL Server instance. With SQL 2000, the default is to support Windows authentication only. This choice should be acceptable for most applications. Before enabling SQL authentication, you should verify that there is no way to provide users access via Windows authentication.

SQL Server and Windows Authentication

Mixed-mode authentication offers support for both the Windows and the SQL Server authentication mechanisms we've discussed. By default, mixed-mode security is *not* enabled. You must enable mixed-mode security one of two ways.

You could enable mixed-mode security during the installation of SQL Server 2000. If you select this option, you would be prompted to enter a password for the sa account. The sa account is the built-in SQL authentication login. When logged in as sa, you have full privileges to the SQL Server, its databases, and all objects within. Because SQL Server's sa account is such a powerful login, it needs a secure password. Previous versions of SQL Server installed with the sa password blank. When installing SQL Server 2000, you must select Mixed Mode to be able to use the sa account at all, and by default you must enter and confirm a password for the sa account. If, for some reason, you decide that you want the sa account with a blank password, you must check the "Blank Password (not recommended)" option. Figure 5.7 shows the option you should select during installation if you want to use both the Windows and the SQL authentication mechanisms.

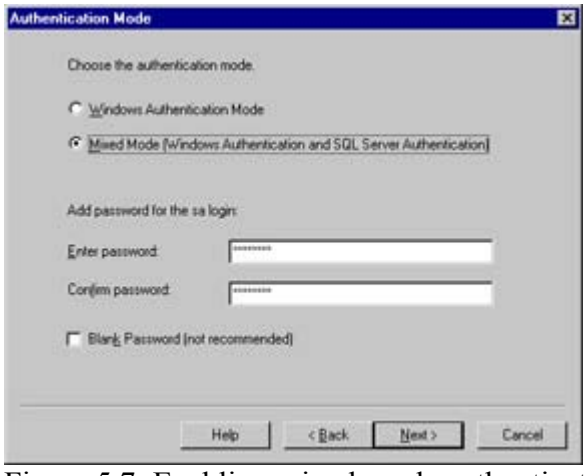


Figure 5.7: Enabling mixed-mode authentication during installation.

Even when SQL authentication is not enabled, you will find that the sa account still appears in Enterprise Manager. Even though the logins are present, they cannot be used if the SQL authentication option is not enabled. If you attempt to log in to a SQL Server using a SQL login but SQL authentication is not enabled, you will receive an error. The error states that the login is "Not associated with a trusted SQL Server connection."

If SQL Server 2000 was installed with the default security mode (Windows only) and you want to change your security mode, you still can. To make the change, use Enterprise Manager. Find your server in the tree, right-click it, and select Properties. The SQL Server Properties window will appear. Switch to the Security tab. On the tab, you will notice two options under the Authentication header: SQL Server and Windows and Windows only. SQL Server and Window" is the option to use for mixed-mode security. Figure 5.8 shows an example of the dialog box with SQL Server and Windows selected.



Figure 5.8: Settings for mixed-mode security.

Changing the authentication mode of your SQL Server requires that the SQL Service be stopped and restarted. When you click OK in the window shown in Figure 5.8 , you will be prompted to restart your SQL Server. If you are ready to restart your server now, answer Yes. Otherwise, answer No, and restart your server when you can. Once the server has been restarted, SQL logins, including the sa account, can be used.

Using Transact SQL Instead of Enterprise Manager

In this chapter, you will be given examples of how to complete administrative tasks using both Enterprise Manager and the Transact-SQL language. If you use only Enterprise Manager to perform administrative tasks, you will be restricted to being able to administer your server only when Enterprise Manager is available.

If you learn the T-SQL statements with which to administer your servers, you can perform administrative tasks via any connection to SQL Server. This includes, of course, the Query Analyzer and OSQL/ISQL tools that come with SQL Server, but it can also mean a connection that you establish to SQL Server in your own VBScript or JScript scripts or any other program you write in VB, VBA, ASP, VC++, and so on.

Windows-Only Authentication Mode

The alternative to mixed-mode authentication is Windows-only authentication. With Windows-only authentication, the SQL authentication mechanism cannot be used, and only Windows logins will have access to SQL Server. We have already discussed numerous security advantages of the Windows authentication mode. One other advantage of limiting logins to Windows logins only is that you have closed down at least one security loophole on your server. With SQL authentication disabled, there is one less path for intruders to take when attempting to gain access to your SQL Server.

Windows-only authentication is the default authentication option for installation SQL Server 2000. If you installed SQL Server otherwise but would like to change the option back to the default Windows-only mode, you can do so rather easily. In Enterprise Manager, right-click your server, and select Properties. When the SQL Server Properties window appears, switch to the Security tab, and under the Authentication header, select "Windows only." You can refer back to Figure 5.8 to see what the security configuration window looks like.

To make the newly selected authentication mode take effect, you need to stop and restart SQL Server. After you click OK in the server configuration window, you will be prompted to restart your SQL Server. If you are prepared for it to restart now, select Yes. Otherwise, if you have users who need to disconnect first, select No, and restart the server when you can.

Logins

There are a number of ways to add new logins to SQL Server. Enterprise Manager provides an easy-to-understand GUI for adding logins and affecting other security settings. However, if you only know how to perform the tasks using Enterprise Manager, you are limited to using only that tool when configuring your server. If you need to be able to configure logins from your own applications or perhaps from a command-line interface, you should understand the T-SQL statements for adding logins. Both methods are demonstrated in this section.

Adding New Windows Logins

Logins control which users can connect; logins can also specify users who should *not* be able to connect. Immediately after installing SQL, some logins already exist. You might want to add logins to allow access to others. You might also require the ability to deny server access to specific individuals or to a group of individuals.

Granting, Revoking, and Denying Windows Logins

As the SQL administrator, you might or might not have administrative privileges in the Windows domain. If you do not have domain admin privileges, you will not be able to create, delete, or modify user or group accounts in the Windows domain. For this reason, you don't "add" or "delete" Windows logins, you *reference* them. The domain administrator needs to create the Windows user or group first. You then reference the existing Windows security object. Allowing a Windows login to connect to SQL Server is called *granting* .

Revoking removes a specific login entry for a Windows account. For example, let's say that the Windows user Martha, a sales department supervisor, was granted login access to SQL Server. At a later time, it was decided to let all the sales supervisors have login access to SQL Server. You ask the domain administrator to create a global group called SalesSupers and have the administrator place all the sales supervisors' Windows domain user accounts into the new group. You then grant the SalesSupers group login access to SQL Server. Martha, however, still has her login specifically granted access. To clean up your security, you should remove the explicit reference to Martha's account. Because she is a sales supervisor, you should have her gain access using that group's login. To remove Martha's explicit login, you would "revoke" it using either Enterprise Manager or Transact-SQL statements. Revoking allows the user to continue to gain login access to SQL Server if he or she is a member of a group that has been granted access.

Denying a login is quite different from revoking it. A user who had his login revoked might still gain access through some group. *Deny* absolutely denies the user login access, even if the user is a member of a group that offers members access. Deny gives us the ability to restrict certain untrustworthy individuals from gaining access to SQL Server, no matter what anybody else says.

Default Logins

There are a few logins that exist on a SQL Server by default. Table 5.1 lists some of the possible logins that exist in your SQL Server immediately after you install it.

Table 5.1: Default SQL Server Logins

Login Name	Login Type	Purpose
BUILTIN\Administrators	Windows Group	Grants login access to any user who is a member of the local Windows Administrators group.
DOMAIN\ServiceAccount	Windows User	If SQLServerAgent is running as an account other than the local system account, the installation routing grants the service account for SQLServerAgent login access to SQL Server. This ensures that the SQLServerAgent can connect to SQL Server to perform automation, maintenance, and replication tasks.

Table 5.1: Default SQL Server Logins		
Login Name	**Login Type**	**Purpose**
sa	Standard	This is the built-in standard SQL account that exists in all servers.

Adding a Windows Login in Enterprise Manager

To add a Windows login using Enterprise Manager, follow these steps:

1. Select your server in the Enterprise Manager tree.
2. Expand the Security node, and select Logins. Figure 5.9 shows the security tree in Enterprise Manager.

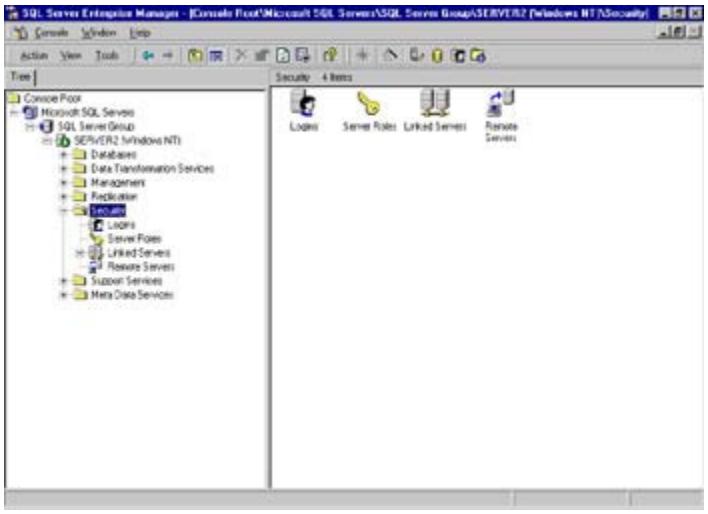


Figure 5.9: The Security tree in Enterprise Manager.

3. In the detail pane to the right, you should see a list of the current logins.
4. Right-click Logins, and select New Login from the pop-up menu. Figure 5.10 shows an example of the SQL Server Login window that appears.

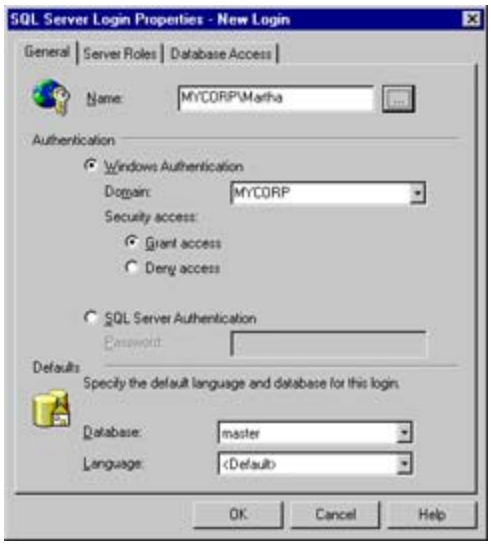


Figure 5.10: Adding a new Windows login.

5. Ensure that the Windows Authentication option is selected.
6. From the Domain drop-down list, select the domain that contains the Windows user.
7. To allow the user to have access to SQL Server, select "Grant access."
8. Select the default database for the user.
9. Click OK.

Rather than typing the username, you can click the "…" button next to the Name field to browse the list of users and groups available in the domain. Recall that a Windows login can reference a Windows user or a Windows group. The process for granting a login to a Windows group rather than a Windows user is identical to the process we've just discussed. Simply select the name of the group to which you want to grant login access, rather than selecting the name of a user.

It should be obvious from Figure 5.10 that to deny a login access rather than grant it, you should simply select the "Deny access" option. Under all circumstances, this option prevents the user (or group of users, if the login is a Windows group) from accessing SQL Server.

To revoke a login using Enterprise Manager, find the login in the list of logins. Highlight it, and delete the entry.

Adding a Windows Login Using Transact-SQL

There is a collection of system-stored procedures for you to use when you are adding or removing Windows logins from Transact-SQL. Table 5.2 lists the system-stored procedures that can be used to manage Windows logins.

Table 5.2: Transact-SQL Statements for Granting, Revoking, and Denying Windows Logins

Stored Procedure	Description
sp_grantlogin 'DOMAIN\UserOrGroupName'	Grants an existing Windows user or group login access to SQL Server.
sp_denylogin 'DOMAIN\UserOrGroupName'	Denies an existing Windows user or group login access to SQL Server.
sp_revokelogin 'DOMAIN\UserOrGroupName'	Revokes a login that has been either explicitly granted or denied using the procedures above.

Consider this example. Martha's user account is the MYCORP domain. For purposes of this example, assume that you have connected to SQL Server with Query Analyzer. To add a login for Martha's account, you would execute the following statement:

```
EXEC sp_grantlogin 'MYCORP\Martha'
```

If you consider the scenario presented in the "Granting, Revoking, and Denying Windows Logins" section, creating the necessary group login and revoking Martha's login could be done as follows:

```
EXEC sp_grantlogin 'MYCORP\SalesSupers'
EXEC sp_revokelogin 'MYCORP\Martha'
```

Remember that even though Martha's login was revoked in the last statement, as long as she is a member of the SalesSupers group in the Windows domain, she still has login access. If you wanted to deny Martha the ability to connect to SQL Server under all circumstances, you would issue this statement:

```
EXEC sp_denylogin 'MYCORP\Martha'
```

After you issued that statement, Martha would no longer be able to log in to SQL Server, even though the group SalesSupers is still granted login access. If a user or a group to which the user belongs is denied login, the user cannot connect—period.

Adding a Standard SQL Login in Enterprise Manager

To add a Windows login using Enterprise Manager, follow these steps:

1. Select your server in the Enterprise Manager Tree.
2. Expand the Security node, and select Logins.
3. In the detail pane to the right, you should see a list of the current logins.
4. Right-click on Logins, and select New Login from the pop-up menu. Figure 5.11 shows an example of the SQL Server Login window that appears.

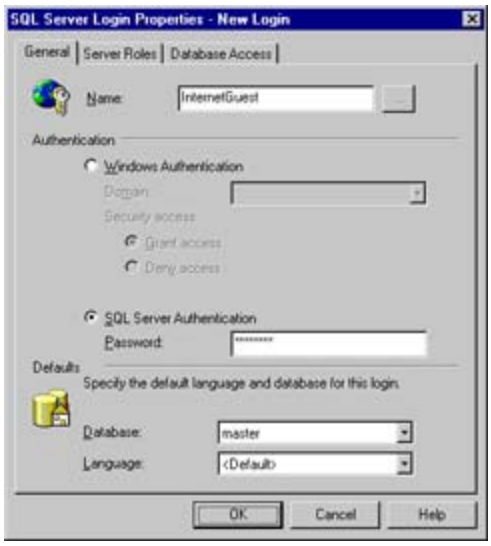


Figure 5.11: Adding a new standard SQL login.

5. Ensure that the SQL Server Authentication option is selected.
6. Enter the name of the SQL login in the Name field.
7. Enter the password for the user in the Password field.
8. Select the default database for the user.
9. Click OK.
10. Confirm the password for the login.

Adding a Windows Login Using Transact-SQL

There is a collection of system-stored procedures for you to use when adding or removing Windows logins from Transact-SQL. Table 5.3 lists the system-stored procedures that can be used to manage standard SQL logins.

Table 5.3: Transact-SQL Statements for Granting, Revoking, and Denying Windows Logins

Stored Procedure	**Description**
sp_addlogin ‘login_name’,’password’,	Creates a new SQL login. ’default database’
sp_droplogin	Drops (or deletes) an existing SQL login.

To add the same login as above using Transact-SQL, you would enter the following:

```
EXEC sp_addlogin 'InternetGuest','password','master'
```

This statement adds a standard SQL login named InternetGuest, with a password of *password* , and a default database of *master* .

To remove this login, you would enter:

```
EXEC sp_droplogin 'InternetGuest'
```

Server Roles

A number of administrative tasks take place at the server level. Creating databases, adding logins, configuring server settings, and the like all require that the individual performing the action be given the permissions to do so. SQL Server uses a collection of server roles (groups that exist at the server level), with predefined permissions to make it easier to grant server-level privileges to individuals.

Fixed Roles

As mentioned before, the easiest way to assign server-level permissions is to make logins members of fixed server roles. The fixed server roles are roles (or groups) that exist inside SQL Server and offer their members permissions to perform certain serverwide actions. Table 5.4 lists all the fixed server roles and identifies the privileges associated with each one.

Table 5.4: Fixed Server Roles

Short Name	Long Name	Description
bulkadmin	Bulk insert administrators	Members can issue the BULK INSERT statement.
dbcreator	Database creators	Members can CREATE, ALTER, and DROP databases; RESTORE databases and database logs; extend databases; and rename databases.
diskadmin	Disk administrators	Members can run the sp_add ump device and sp_dropdevice stored procedures.
processadmin	Process administrators	Members can use the KILL statement to break user connections to SQL Server.
securityadmin	Security administrators	Members have a number of security assignment privileges. Security administrators can grant, revoke, or deny the CREATE DATABASE permission. Security administrators can add and drop SQL logins; grant, revoke, or deny Windows logins; and add linked server and remote server logins.
serveradmin	Server administrators	Members can issue SHUTDOWN and RECONFIGURE statements; run the sp_configure, sp_tableoption, and sp_fulltext_service stored procedures; and free procedure cache buffers.
setupadmin	Setup administrators	Setup administrators can add, drop, and configure linked servers. They also mark a

Table 5.4: Fixed Server Roles		
Short Name	**Long Name**	**Description**
		stored procedure as a startup procedure.
sysadmin	System administrators	Members of this role can perform any server action. Once a login is a member of this role, it needs no other role membership or permissions to have full access to SQL Server, its databases, and all objects within those databases.

Adding a Login to a Fixed Server Role in Enterprise Manager

Figure 5.12 shows an example of adding the MYCORP\SalesSupers login, created previously, to the sysadmin role. To get the window shown in Figure 5.12 , you right-click the login in Enterprise Manager and select Properties from the pop-up menu. When the window appears, select the Server Roles tab, and click the check box next to the role of which you want to make the login a member.

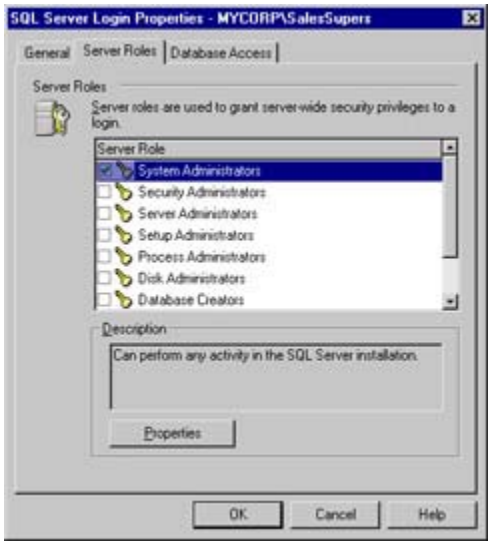


Figure 5.12: Fixed server role membership.

Adding a Login to a Fixed Server Role Using Transact-SQL

Logins can be added to fixed server roles using Transact-SQL as well as Enterprise Manager. There are a number of stored procedures you can use when working with fixed server roles and their membership. Table 5.5 lists the system-stored procedures for managing fixed server role memberships.

Table 5.5: Transact-SQL Statements for Managing Fixed Server Roles	
Stored Procedure	**Description**

Table 5.5: Transact-SQL Statements for Managing Fixed Server Roles

Stored Procedure	Description
sp_addsrvrolemember 'login', 'role'	Adds logins to a server role.
sp_dropsrvrolemember 'login', 'role'	Removes a login from the server role.
sp_helpsrvrole	Returns a basic list of the fixed server roles.
sp_helpsrvrolemember 'role'	Shows a list of the members of the specified role.
sp_srvrolepermission 'role'	Lists the permissions each specified role offers its members.

To add the MYCORP\SalesSupers login to the sysadmin role using the T-SQL statements, you would execute the following command:

```
EXEC sp_addsrvrolemember 'MYCORP\SalesSupers' ,'sysadmin'
```

Database Users

Every database has at least one user account in it. The user account is named dbo, which stands for *database owner* . The dbo user account maps to the login of the user who "owns" the database. Normally, the owner is the user account of the login who created the database in the first place. If a member of the sysadmin role made the database, all sysadmins are the "owners" of the database. If, however, a specific user was given the create database permission (by being made a member of the dbcreators fixed database role, for example), the databases that the user creates are owned by that user. That username inside the databases he or she owns is dbo. For example, say that the user MYCORP\Frank is given create database permissions. Frank creates a database called Sales. Since Frank created the database, he owns it. His login name, MYCORP\Frank, has a user account in the Sales database. The user account is named dbo. Whenever Frank is in the Sales database, he is known as dbo.

If Frank were to leave the company, you might want to reassign ownership of his database to yourself or another login. To change the login who owns a database, you can use the sp_changedbowner stored procedure. Figure 5.13 shows the user information for the Northwind database as viewed in Enterprise Manager.

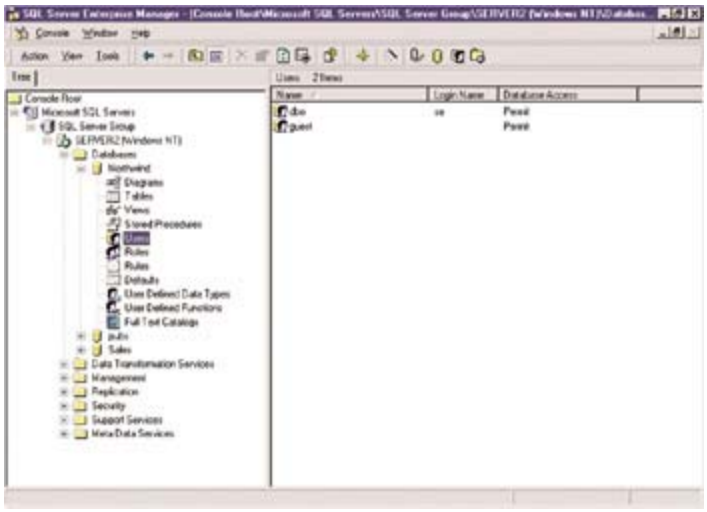


Figure 5.13: Northwind database users.

Adding New Database Users

The logins we created allow only the people or group of people they represent to connect to SQL Server. Logins do not specify to which databases people have access. To allow the individuals represented by a login to gain access to a given database, a user account must be created for that login in the database.

Adding a User Account to a Database in Enterprise Manager

To add a user account to a database in Enterprise Manager, follow these steps.

1. Expand the node for the database in Enterprise Manager.
2. Select the Users node. A list of current users should appear in the detail pane.
3. Right-click Users, and select New Database User. Figure 5.14 shows the window that appears.

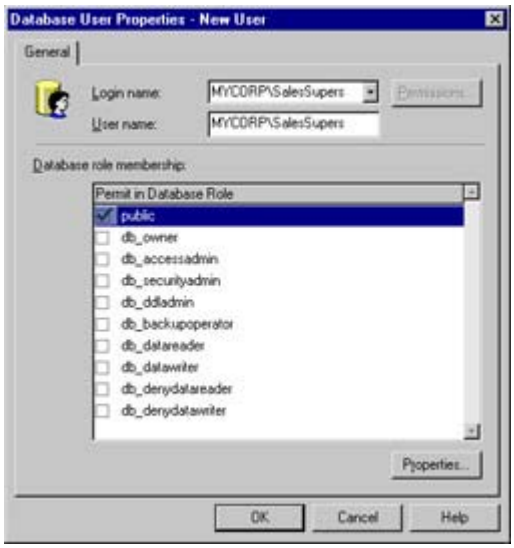


Figure 5.14: Database user properties.

4. Select the login that needs access to the database from the "Login name:" drop-down list.
5. Confirm that the username is the same as the login name (recommended).
6. Click OK.

Adding a User Account to a Database Using Transact-SQL

Users can, of course, be added using T-SQL. Table 5.6 highlights the key system-stored procedures you would use to add and drop users to and from a database.

Table 5.6: Transact-SQL Statements for Granting and Revoking Database Access

Stored Procedure	Description
sp_grantdbaccess 'login','username'	Creates a new database user account for a given login.
sp_revokedbaccess 'username'	Revokes a user account from the database.

For example, to grant the MYCORP\SalesSupers login access to the Northwind database, you would issue the following statements:

```
USE Northwind
EXEC sp_grantdbaccess 'MYCORP\SalesSupers', 'SalesSupers'
```

To remove access from the database:

```
USE Northwind
EXEC sp_revokedbaccess 'SalesSupers'
```

The Guest User Account

As mentioned earlier, a database user account maps to a login. There could be situations in which you want to grant any and every login access to a given database. Rather than having to grant database access to each login as it is created, you can create a guest user account in the database. The guest account in a database is used when the individual who is attempting access has a server login but not a database user account.

Because the guest login maps to all unspecified logins rather than any specific login, you create it a little differently from a normal login. The statement to add the guest account to a database is:

```
EXEC sp_grantdbaccess 'guest'
```

Notice that unlike previous use of the sp_grantdbaccess procedure, only the username is supplied. Since *guest* really maps to all logins, you do not specify a login when adding the guest account.

Once the guest account is added to a database, all logins that can connect to SQL Server can access the database. You can review the security in the Northwind and pubs databases the see how the guest account has been implemented. You might also notice that the guest account has been added to the master, msdb, and TempDB databases. This is because in general, all logins need access to those databases in one form or another. In fact, although you can remove guest access from the msdb database, you cannot remove it from master or TempDB. Master contains certain metadata that is required by all logins, and TempDB is a necessary storage mechanism that all logins might need to employ.

Assigning User Permissions

Permissions need to be assigned for users to be able to execute statements and interact with the objects in a database. Permissions indicate the statements and operations users and roles in a database are allowed to perform. Each type of object has its own set of permissions that can be granted, revoked, or denied. Some permissions are given at the server level, offered by fixed server roles; some permissions are given at the database level, such as CREATE TABLE and CREATE VIEW; and some permissions are given at the object level, including SELECT, INSERT, and EXEC. Table 5.7 lists the permissions that can be assigned at various levels in SQL Server.

Table 5.7: Assignable Permissions

Assignable To	Permission	Description
Statements executed at the server level	CREATE DATABASE	Actually assigned in the master database, this permission allows its recipient to create databases on the server. A preferable way to assign the permission is to make the intended recipient a member of the dbcreator fixed server role.

Table 5.7: Assignable Permissions

Assignable To	Permission	Description
Statements executed at the database level	CREATE TABLE CREATE VIEW CREATE PROCEDURE CREATE DEFAULT CREATE RULE CREATE FUNCTION	These permissions allow users to create related objects. By default, only members of the sysadmin, db_owners, and ddl_admin roles can issue these statements.
	BACKUP DATABASE BACKUP LOG	Allows the grantee to back up the database and the transaction log.
Tables and views	SELECT	Allows users to read data in a table or view.
	INSERT	Allows users to insert data into a table or view.
	UPDATE	Allows users to update data in a table or view.
	DELETE	Allows users to delete data from a table or view.
	REFERENCES	Allows users to create objects that reference (as in a foreign key/primary key relationship) the specific object.
Columns	SELECT	Allows users to select from that column only.
	UPDATE	Allows users to update that column only.
Stored Procedures	EXEC	Allows the user to execute a stored procedure.
User-Defined Functions	SELECT	Allows a user to execute the function and retrieve data.
	REFERENCES	Allows a user to define an object that references the function (as in a foreign key/primary key relationship).
Unassignable	Fixed server role permissions	Fixed server roles offer the ability to grant permissions that are otherwise unattainable to users. The only way to offer the permission is to make the login a member of the fixed server role.
	Fixed database role permissions	As with fixed server roles, fixed database roles can offer permissions that are not available any other way. For example, to execute CREATE INDEX statements, you must be a sysadmin, db_owner, or ddl_admin.
	Creator/owner	When a login creates a database or a user

Table 5.7: Assignable Permissions

Assignable To	Permission	Description
	permissions	creates an object, he or she owns that database or object. As the owner, that person has full control over the object, including the ability to grant permissions to other users.

The statements used to specify permissions are GRANT, REVOKE, and DENY. The concepts of granting, revoking, and denying are the same as with login access. Granting access says, "Yes, you can." Revoking says, "I won't say that you can specifically, but if some other group lets you do it, fine." Denying says, "No way, nohow, can you perform this action, no matter what any other group or user allows you to do." Table 5.8 lists the T-SQL statements used to grant, revoke, and deny permissions.

Consider these examples. To grant the SELECT permissions on the Products table to the InternetGuest user account, you would issue the following statement

```
GRANT SELECT ON Products TO InternetGuest
```

To revoke the permission for InternetGuest to EXECUTE the CustOrderHist stored procedure, you would issue the following statement:

```
REVOKE EXECUTE ON CustOrderHist FROM InternetGuest
```

To deny the public role and the guest user SELECT, INSERT, UPDATE, and DELETE permissions on the Employees table, you could execute the following:

```
DENY SELECT, INSERT, UPDATE, DELETE ON Employees TO [public],guest
```

Of course, permissions can be assigned using Enterprise Manager as well as T-SQL. To get to the permissions for an object in Enterprise Manager, highlight the object in the tree, right-click it, and select Properties from the pop-up menu. In the Object Properties window that appears, click the Permissions button. Figure 5.15 shows an example of the Permissions window with various permissions set for the Customers table in Northwind. A checked box for a permission implies that the permission is granted. A box with an X in it implies that the permission has been denied, and a cleared check box means that the permission is revoked.

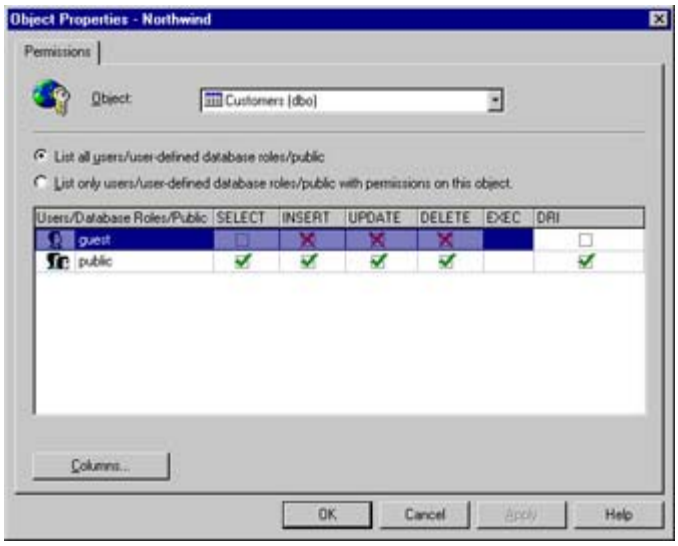


Figure 5.15: Setting permissions in Enterprise Manager.

Database Roles

Assigning permissions to groups of users as a whole is preferable to assigning permissions individually to each user who needs them. This rule holds true for permissions in your databases. Consider an application that contains hundreds or even thousands of objects. Many popular enterprise resource planning (ERP) solutions in use today contain literally thousands of tables. It would be impossible to reliably set permissions to the tables and other objects in those databases on a user-by-user basis. Instead, if groups (or in SQL terms, roles) were created and permissions assigned to the roles, users could be added to the roles and automatically receive the permissions assigned to the roles.

Table 5.8: Transact-SQL Statements for Granting, Revoking, and Denying Permissions

Statement	Description
GRANT permission list ON object TO users and roles	The permissions list is a comma-delimited list of permissions that are o be granted to a specific object to a comma-delimited list of users and roles.
REVOKE permission list ON object FROM users and roles	Revokes the specific assignment of the permissions from the users and roles listed.
DENY permission list ON object TO users and roles	Specifically denies the listed users and roles performing the actions named in the permission list.

SQL Server offers two classes of roles in a database: fixed database roles and user-defined roles. Similar to fixed server roles, the fixed database roles are predefined, cannot be deleted, and (other than the public role) cannot have their permissions changed. The only thing you can change about fixed server roles is their membership.

User-defined roles allow you to create your own roles for your own purposes. You assign permissions to a role according the needs of the role's intended membership, and you fill the role with members.

Fixed Roles

There are a number of fixed database roles. These roles can help you assign permissions to users that otherwise would be unassignable. Some fixed server roles offer a convenient way to grant blanket permissions to all objects in the database.

Table 5.9 lists all the fixed server roles and identifies the privileges associated with each one.

Table 5.9: Fixed Database Roles

Short Name	Description
db_owner	Members can perform any action in the database. If membership to this role is granted, no other membership or permission is required for the user.
db_accessadmin	Members can grant and revoke database access and alias user accounts to logins.
db_ddladmin	Members can create any object in the database. They can run DBCC CLEANTABLE, SHOWCONTIG, and SHOW_STATISTICS. In addition, members can change table options, truncate tables, rename objects, change object owners, and manage full-text indexing on tables.
db_backupoperator	Members can issue the BACKUP DATABASE, BACKUP LOG, and CHECKPOINT statements.
db_securityadmin	Members can manage object permissions; create, modify, or drop roles; and manage role membership.
db_datareader	Allows members to select from any table, view, or user-defined function.
db_datawriter	Allows members to insert, update, or delete data in tables and views.
db_denydatareader	Denies the ability to select (read) any data from tables, views, and user-defined functions.
db_denydatawriter	Denies the ability to insert, update, or delete data in any table or view.
public	Every user who enters a database is part of the public role. Public provides a convenient way to assign permissions to all users in a database. By granting permissions to public, you grant permission to all users in the database. You need to be careful about what you GRANT and what you DENY to the public role.

Users can be added to roles in Enterprise Manager by simply highlighting the user account, right-clicking it, and selecting Properties from the pop-up menu. In the Database User Properties window that appears, select or deselect the appropriate roles from the list of roles. Figure 5.16 shows an example of the Database User Properties window.

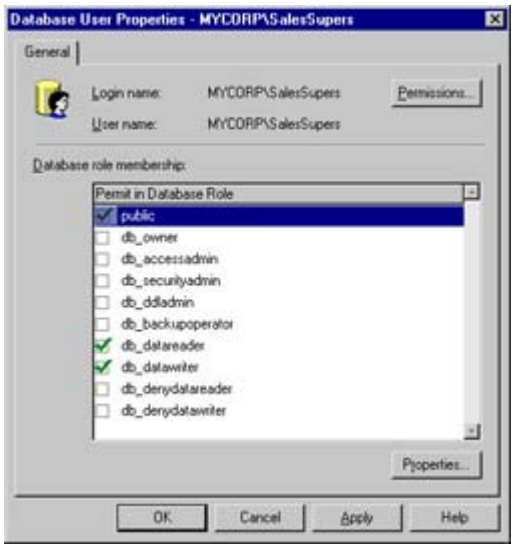


Figure 5.16: Changing a user's role membership.

To add users to roles in Transact-SQL, you can use a few stored procedures. In the list shown in Table 5.10 , note that only one of the procedures is specific to fixed database roles. All other procedures work the same, whether you are working with a fixed database role or a user-defined role. Table 5.10 lists the system-stored procedures that are used to manage database role memberships.

Table 5.10: Transact-SQL Statements for Managing Database Roles

Stored Procedure	**Description**
sp_addrolemember 'role', 'user'	Adds a user to a database role.
sp_droprolemember 'role', 'user'	Removes a login from a database role.
sp_helprole	Returns a basic list of all database roles, including fixed database roles.
sp_helprolemember 'role'	Shows a list of the members of the specified role.
sp_dbfixedrolepermission 'role'	Lists the permissions each specified role offers its members.

User-Defined Roles

If the fixed database roles do not offer you enough segmentation of users and permissions, you can create your own roles, for whatever reasons you like. Two flavors of roles can be created in a database. *Standard roles* are roles to which users are added

ahead of time. Generally, you add users to standard roles when the role is created or when a new user is added to the database. These memberships are *persistent* , meaning that even if the user disconnects or reconnects to the database or even if the SQL Server is restarted, the role's membership remains unchanged.

Application roles , the other kind of role that can be created in a database, are similar to other roles in that they are assigned permissions, and the users who are placed into the roles assume those permissions. The differences are when a user is added to an application role, how the user is added to the role, and how long the user stays in the role.

Standard Roles

Standard database roles allow you to create roles of your own, assign permissions to them, and add members. With a standard role, the role is created, permissions assigned, and members added or removed at "administration" time. This means that these tasks are done as part of your normal administrative tasks. As new users need to be added to a role, you manually add them. To remove users, you manually remove them.

Standard roles can be created for whatever purpose you like. Any given user in a database can belong to one or more roles. In fact, as stated previously, every database user belongs to at least one roll because all users are members of *public* .

To create an application role in Enterprise Manager, expand the database that is to contain the role. Right-click the Roles node, and select New Database Role from the pop-up menu. Figure 5.17 shows an example of adding a new role called BasicAccess to the database.

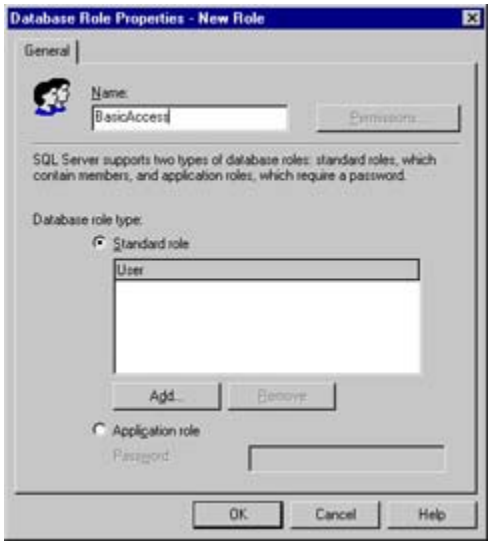


Figure 5.17: Adding a new database role.

Of course, T-SQL statements can be used to create and drop user-defined roles as well. Table 5.11 lists the system-stored procedures that can be used to manage user-defined roles.

Table 5.11: Transact-SQL Statements for Creating and Deleting Standard Database Roles

Stored Procedure	Description
sp_addrole 'rolename'	Creates a standard role in the database.
sp_droprole 'rolename'	Drops (or deletes) a standard role from the database.
sp_helprole	Returns a basic list of all database roles, including fixed database roles.

For example, to add the role BasicAccess using the stored procedures instead of Enterprise Manager, you would execute the following statement:

```
EXEC sp_addrole 'BasicAccess'
```

For adding and removing members from standard database roles, use the same stored procedures for adding and removing members from fixed database roles that were documented in the previous section .

Once the standard role is created, you would use the GRANT, REVOKE, and DENY statements to assign permissions to the role exactly the same way you would have assigned permissions to a database user.

Application Roles

Earlier you read that application roles differ from standard roles in terms of how their membership is defined and how long a user remains a member of the role. Application roles are still roles, however. You create application roles and assign permissions to the role, and those permissions are automatically bestowed on the role's members. The question is, how and when do we put users into the role?

Application roles somewhat resemble a secret club. Like any good secret club, there is a password to get in. When you create an application role, rather than assigning its membership ahead of time (at "administration" time, as we called it earlier), you put a password on the role. For a user to become a member of the role, the user must supply the proper password. Once the correct password has been supplied, the user stays in the role for the life of the connection. However, as soon as the user closes the connection, he or she is removed from the role and to re-enter the role must resupply the role's password each time he or she connects.

You don't tell the *user* that the role exists, let alone what its password is. Instead, you provide that information to the developer of the client application. The client application allows the user to connect to the database using his or her normal credentials. Once the user has successfully connected, the application activates the application role in the background, giving the user the permissions needed to work with the data. When the user closes the application, the connection is broken, and the user is no longer a member of the role.

If the user were to attempt to connect to the database using a tool other than the approved client application—Query Analyzer, perhaps—the user would be able to connect but would be unaware that an application role exists. Even if the user knew about the application role, he or she would not know the password to activate the role and gain temporary membership.

To create an application role, you can use either Enterprise Manager or T-SQL statements. To create an application role in Enterprise Manager, follow the same steps as you would to create a standard application role. Just pick the option for the application role, and enter the password. Figure 5.18 shows an example of creating an application role named OrderEntryAppRole.



Figure 5.18: Creating a new application role.

Notice that in Figure 5.18 , once the "Application role" option is selected, the "Add" button is grayed out. This happens because you cannot define an application role's membership at administration time. The only way for members to get into the role is to have the role activated with the correct password once they are connected to an application. You cannot activate an application role using Enterprise Manager. Role activation takes place in the client application after the user has connected.

As was stated earlier, there are T-SQL stored procedures you can use to create and activate application roles, as displayed in Table 5.12 .

Table 5.12: Transact-SQL Statements for Creating, Deleting, and Activating Application Roles

Stored Procedure	**Description**
sp_addapprole 'rolename','password'	Adds an application role and assigns the activation password.
sp_dropapprole 'rolename'	Drops (or deletes) an application role from

Table 5.12: Transact-SQL Statements for Creating, Deleting, and Activating Application Roles

Stored Procedure	Description
	the database.
sp_setapprole ‘rolename’,’password’	Puts the current user into the specified application role, as long as the password is correct.

The following example statement shows how you could create the application role OrderEntryAppRole using stored procedures:

```
EXEC sp_addapprole 'OrderEntryAppRole','rolepass'
```

Once the role is created, you can use the standard GRANT, REVOKE, and DENY statements to configure its permissions. Then when a user connects using your application, your application would issue the following statement on the user’s behalf and activate the application role for the life of the connection. An example of the statement the client application would issue follows:

```
EXEC sp_setapprole 'OrderEntryAppRole','rolepass'
```

Implementing Database and Server Security

This section applies the concepts discussed in the chapter to a hypothetical situation. The goal is to demonstrate how the various levels of security would be implemented in a real-world situation.

The Scenario

As we review the security options we've defined, consider the scenario presented in Figure 5.19 . MyCorp, the company you work for, has a sales department. MyCorp installed Windows 2000 with Active Directory and created an Active Directory domain named MyCorp to contain its user accounts and groups. The sales users in the sales department are divided into two different types: normal sales staff users and sales supervisors. Both types of user need to be able to connect to SQL Server and access data in the Northwind database.

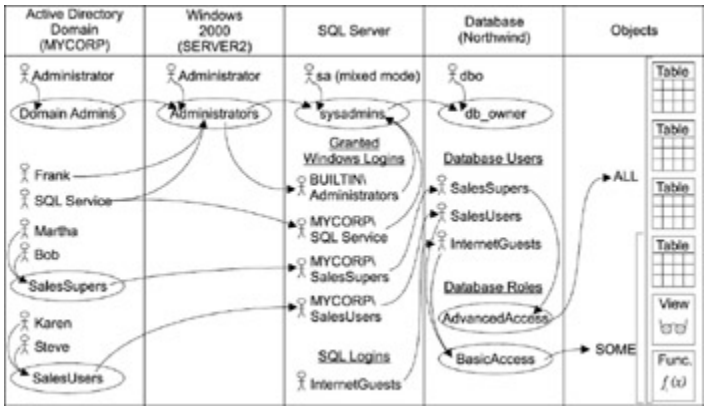


Figure 5.19: A security scenario.

In addition to the sales staff, you have a Windows user named Frank who is the SQL Server DBA responsible for this specific SQL Server. Frank is not a domain administrator, but he does need administrative rights on the Windows 2000 server where SQL Server is installed.

The SQL Server service will be running as a domain account so that it can interact with other services (such as Exchange) on the network. You have decided that the SQL Server service account will have administrative access to the SQL Server itself. The domain administrator has already created an account named SQLService for you to use as the service account for SQL Server.

Along with the existing Windows users, some external sales reps will be connecting into the database via the Internet for some basic needs. Some of these users run Windows on their client machines, others run Apple systems, and still others use Linux. You have decided to create a standard SQL login named InternetGuests to provide login access to those users.

You need to create the logins, user accounts, roles, and permissions to effectively implement the scenario presented in Figure 5.19 .

User Authentication

Securing the database starts at the server level. For the database itself to be secure, the operating system on which SQL Server runs must also be secure. To start locking things down, you will want to first secure the operating system and then move on to controlling access to SQL Server itself.

Operating System Administrative Access

As stated earlier, the SQL Server service will be running as a domain account that has already been created in the domain. You need to ensure that the service account is a local administrator on the system that will be running SQL Server. In addition, the account used by SQL Server, Frank's user account, will also need administrative access to the system because he will be responsible for maintaining this server. You can use the standard Windows 2000 "net" command-line tool to make sure that both these account permissions are configured. The following statements, run at the command prompt on the Windows 2000 server on which SQL runs, will ensure those administrative permissions for Frank and the SQLService service account:

```
NET localgroup Administrators MYCORP\Frank /add
NET localgroup Administrators MYCORP\SQLService /add
```

Windows 2000

Recall that SQL Server 2000 supports two authentication modes. The Windows-only authentication mode provides support for only Windows logins. Our scenario has a number of Windows users and groups who will need access to SQL Server, but we also need to provide support for the InternetGuest standard account that will be used by Internet-based clients. In this situation, the Windows-only authentication mode is too restrictive, so mixed-mode authentication needs to be enabled. If all users who required access to the server had pre-existing Windows accounts, we could accept the default option of Windows authentication. In this example, you must make sure that your authentication mode is set to support both the SQL Server and Windows authentication options.

A few default logins exist in SQL Server after a normal installation. The BUILTIN\Administrators group from the local system is one of those default logins. The BUILTIN\Administrators group represents the administrators group on the NT system on which SQL Server runs. Since the Frank and SQLService user accounts were made members of the local administrators group, both of those accounts now have access to SQL Server via the BUILTIN\Administrators login. When the BUILTIN\Administrators login was created during the installation process, it was also made a member of the sysadmin fixed server role in SQL Server, meaning that the users represented by it have full access to SQL Server.

During installation, the SQLService user account was used by the SQLServerAgent. The SQLServerAgent's job is to automate routine tasks in SQL Server, assist in database maintenance, and execute replication tasks. To do its job, the SQLServerAgent must be able to connect to SQL Server. To make sure this happened, the service account was added by default during the installation of the SQL Server 2000 server. This login is also added to the sysadmin fixed server role to gain full access to SQL Server.

The final default login that exists is the sa account. You should ensure that the sa account has a secure password and, having done so, never use the account again. As long as your Windows users can log in to SQL Server with administrative access, there is no need to use sa.

When it comes to adding the logins for the sales users and sales supervisors, it is assumed that the domain administrator has already created these global groups in the Active Directory domain and placed the appropriate users into those groups. To add the logins to SQL Server, execute the following commands using the SQL Query Analyzer tool:

```
EXEC sp_grantlogin 'MYCORP\SalesUsers'
EXEC sp_grantlogin 'MYCORP\SalesSupers'
```

SQL Server Logins

SQL Server logins provide you with an option to allow non-Windows users access to your SQL Server. In this scenario, the Internet clients who will access SQL data are coming from a variety of operating systems and platforms. To support login access, you need a login mechanism that all of them support. The InternetGuest account you will implement is a Standard SQL login, which ensures that the users can connect with the account, regardless of OS, Web client, or the like they connect with.

To make sure that SQL Server logins are accepted, you should confirm that the SQL Server's authentication mode is set to support SQL Server and Windows authentication mechanisms. Finally, to add the standard SQL login InternetGuests with the password *guestpass* , execute the following command using the SQL Query Analyzer tool:

```
EXEC sp_addlogin 'InternetGuests','guestpass'
```

Assigning Permissions

Once the logins have been added, the next step is to create user accounts in the database, create the database roles, assign permissions to those roles, and finally, add the users to the appropriate roles.

Adding Users to Database Roles

The users can now connect to SQL Server. You must further define their access by creating user accounts and roles in the Northwind database. To add the user accounts to the database, you would issue the following statements:

```
USE Northwind
EXEC sp_grantdbaccess 'MYCORP\SalesUsers','SalesUsers'
EXEC sp_grantdbaccess 'MYCORP\SalesSupers','SalesSupers'
EXEC sp_grantdbaccess 'InternetGuests','InternetGuests'
```

Rather than assigning permissions directly to user accounts, you should create roles and assign the roles' permissions to the objects in the database. In this scenario, the database users are divided into two basic groups: users who need basic access to the data objects users who need advanced access to the database objects. To create the two roles to represent these groups of users, execute the following statements:

```
EXEC sp_addrole 'BasicAccess'
EXEC sp_addrole 'AdvancedAccess'
```

Finally, add the appropriate users to the roles you just created:

```
EXEC sp_addrolemember 'BasicAccess','SalesUsers'
EXEC sp_addrolemember 'BasicAccess','InternetGuests'
EXEC sp_addrolemember 'AdvancedAccess','SalesSupers'
```

Assigning Permissions to Users and Roles

To finish things up, you need to assign permissions to the roles so that their members can access the data. The following statements grant some basic permissions to certain database objects in the Northwind database to the BasicAccess role:

```
GRANT SELECT ON Products TO BasicAccess
GRANT SELECT ON Customers TO BasicAccess
GRANT SELECT ON Orders TO BasicAccess
GRANT SELECT ON [Order Details] TO BasicAccess
GRANT EXEC ON [Ten Most Expensive Products] TO BasicAccess
```

For the members of the AdvancedAccess role, more comprehensive permissions are required:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON Products TO AdvancedAccess
GRANT SELECT, INSERT, UPDATE, DELETE ON Customers TO AdvancedAccess
GRANT SELECT, INSERT, UPDATE, DELETE ON Orders TO AdvancedAccess
GRANT SELECT, INSERT, UPDATE, DELETE ON [Order Details]TO
AdvancedAccess
GRANT EXEC ON CustOrderHist TO AdvancedAccess
GRANT EXEC ON [Ten Most Expensive Products] TO AdvancedAccess
GRANT EXEC ON [Sales By Year] TO AdvancedAccess
```

Network Communications Security

For the clients who will be connecting to SQL Server across the Internet, you need to consider the encryption of data as they travel across public networks. You have a number of options available to you.

Multiprotocol Encryption

For users who might be connecting with older clients, you can maintain compatibility with them while still enabling encryption by using the multiprotocol network library. The multiprotocol network library became popular because it supports client/server communication over a variety of network protocols (TCP/IP, NWLink, and NetBEUI). It uses the Microsoft Remote Procedure Call (RPC) Interprocess Communication Mechanism (IPC). Microsoft RPCs have an encryption capability that makes it possible for SQL DBAs to encrypt SQL traffic as it moves from client to server and back. It was the only network encryption option available to SQL Server until the latest version. You could, of course, use VPN solutions such as PPTP to establish secure connections and send traffic through those VPNs, but that was completely outside the scope of SQL Server itself.

One downside to using the multiprotocol network library is that as the client connects to use Microsoft RPCs, the client computer itself must establish a connection with the SQL Server. This requires that the user be logged in using a valid Windows account that can be recognized by the SQL Server. This will eliminate non-Windows remote clients from being able to use multiprotocol to connect, even if you have created a standard SQL login for them to use. One other restriction is that the multiprotocol network library does not support connecting to a named instance of SQL Server 2000.

Warning Enabling encryption of any kind secures your network traffic. It also increases communications overhead due to the encryption and decryption processes. You will find that with encryption enabled, whether it's multiprotocol, SSL, or IPSec, network traffic will be increased and, to some degree, response times will increase. Although these factors result in minimal increases in response time and resource use, you should be aware of the pitfalls of encryption versus the advantages of added security and protection of your data.

Configuring Multiprotocol Encryption

If you have remote users who have valid Windows accounts and need to connect only to the default instance of SQL Server, the multiprotocol network library can be used effectively. To enforce encryption of all multiprotocol traffic, configure the protocol on the server to enable encryption. You must first make sure that the multiprotocol library is enabled. Figure 5.20 shows the Server Network Utility with the multiprotocol network library enabled.

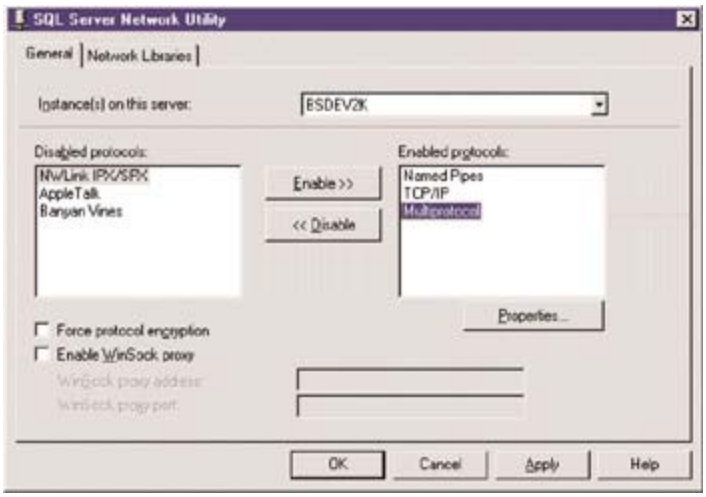


Figure 5.20: Server Network Utility with the multiprotocol library enabled.

To enable encryption for all multiprotocol communication, select the Multiprotocol item under "Enabled protocols:" and click Properties. The Multiprotocol Properties window will appear. Check the box "Enable encryption" to force all multiprotocol network library traffic to be encrypted. Figure 5.21 shows the Multiprotocol Properties window with encryption enabled.

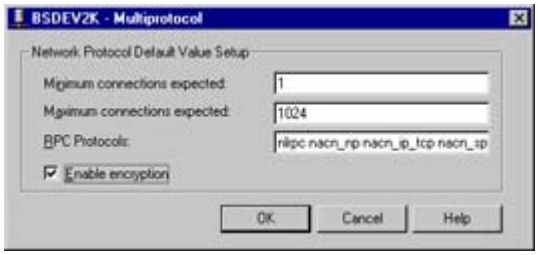


Figure 5.21: Enabling Multiprotocol Encyrption.

SSL Support

SSL is an existing security technology that has received wide acceptance on the Internet. The same SSL technology that has been used to secure Web communications for years can now also be used to secure traffic to and from SQL clients.

Configuring SSL Support for SQL Server

New with SQL Server 2000 is the ability to encrypt data using SSL, a standard technology for encrypting data using public and private key encryption methods. SSL has been in use for years as a way to enable secure communications with Web servers, mail servers, and so on. You, as the administrator of your SQL Server, now have the ability to use SSL to encrypt your SQL traffic as it moves from client to server and back.

With enabled SSL on the server, you need to first acquire a valid SSL key from a "recognized" certificate authority (CA). "Recognized" means that it must be trusted by both the server and the clients. There are a number of commercial CAs out there you

might have heard of; VeriSign as probably the most popular one. However, Microsoft Windows 2000 ships with its own Certificate services, and you can become your own CA for with intranet and extranet solutions. The configuration of those services and the request and installation of SSL keys are beyond the scope of this section. For further information on these topics, read up on Certificates in the Windows 2000 Books Online.

Once you have acquired a certificate from a recognized CA and installed it on the system that is running SQL Server 2000, SQL Server can now use that certificate to force encryption of all traffic between client and server.

This encryption is different from the multiprotocol encryption discussed earlier. SSL encryption is available on *all* network libraries. It also works independently of the Windows authentication services and can encrypt traffic to and from any SQL Server 2000 client. To enable encryption of all traffic on SQL Server, use the Server Network utility. Check the "Force protocol encryption" option to ensure that all client/server traffic is encrypted. Figure 5.22 demonstrates the Server Network utility with this option checked.

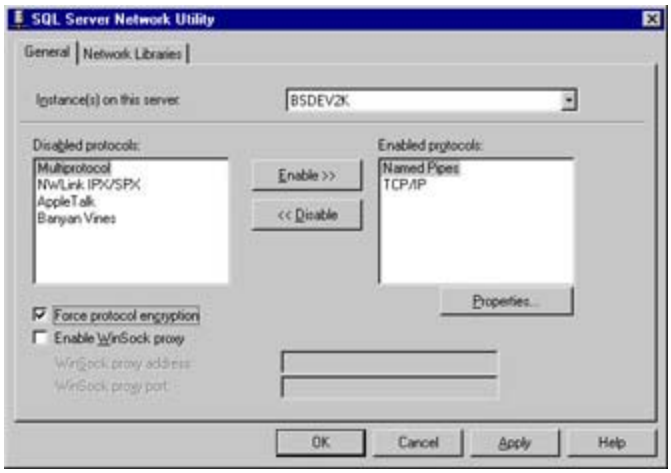


Figure 5.22: Forcing protocol encryption.

Tip To get a test SSL key, use the Internet Services Manager snap-in in the MMC to create a certificate request. You can submit that certificate request to VeriSign to receive a 14-day trial SSL that can be use for testing purposes. When you receive the key back from VeriSign, install it using the Internet Services Manager. Once the SSL key is successfully installed, you can check the "Force protocol encryption" option in the Server Network Utility to test SSL encryption. To find out more about VeriSign's trial IDs, go to http://digitalid.verisign.com/ .

IPSec in Windows 2000

Windows 2000 offers an extremely configurable network security technology called IPSec. With IPSec, the network administrator can configure network security that has varying levels of encryption, depending on the level of the organization at which a computer or user is, the destinations with which they are communicating, and the

protocols used. IPSec offers a comprehensive network security solution that is integrated with the operating system and with Active Directory.

Because IPSec is part of the OS services, SQL Server itself requires no specific knowledge of IPSec or its configuration. SQL Server merely uses standard OS calls to send and receive data via the network, and if IPSec is used, the security policies and authentication mechanisms happen transparently. For further information, look up Internet Protocol Security (IPSec) in the Windows 2000 Help.

Summary

In this chapter, you reviewed the options and solutions that can be used to secure your SQL Server 2000 servers. SQL Server security, although involving many levels, technologies, and methods, starts with a plan. When you plan your SQL Server security model, you must first think about securing the OS and file system on which SQL Server will be installed. Locking down physical and network access to the OS will ensure that malicious users don't bypass SQL Servers security completely and pull the rug out from under your SQL Server by deleting or stealing its files right off the server.

Once the operating system is secure, you need to cautiously grant login access to SQL Server. Login access can be granted to existing Windows users or groups. The beauty of this solution is that you do not create new users or groups, you instead reference existing users and groups and "grant" them access to connect. If you have users who are connecting from non-Windows systems or from clients outside your Windows domain, you can allow them to connect to SQL Server using standard SQL logins that SQL Server authenticates.

Database access is secured by creating user accounts for each login that needs access to a given database. The separation of a login and its database access gives you, the SQL administrator, the ability to let some SQL logins access certain databases while other SQL logins access other databases. In other words, you have flexibility.

Permissions need to be assigned to allow individuals to work with the server, its databases, and the objects and data within them. You could assign permissions on a user-by-user basis, but this can become tedious, if not impossible, as the numbers of users and objects increase. A preferred method is to use either the built-in fixed server roles and fixed database roles to assign permissions or create your own roles at the database level, assign permissions to the roles, and place the users that need permissions into the appropriate roles.

SQL Server itself might be secure as Fort Knox, but if the traffic—the communication, in other words—between the clients and the server is not secure, you have not done yourself much good. You can encrypt data as they travel from client to server using a variety of solutions with SQL Server 2000. Multiprotocol and SSL encryption can be configured at the SQL Server to ensure secure communications, or you might decide to work with the Windows 2000 and Active Directory administrators to configure IPSec as a more complete and comprehensive network security solution.

If the security plan is well thought out and implemented, modifying existing permissions and adding and removing users from the system become almost trivial tasks. Spend time planning and implementing the security model, and your day-to-day administrative load will be significantly reduced.

FAQs

Q. I have deleted the Windows login from my SQL server for a user who moved to a different department, but the user can still connect to SQL Server. Why?

A. Remember that Windows users can connect to SQL Server if their user accounts have been granted login access or if a group of which the user is a member has been granted login access. You need to review the Windows groups that have been granted login access to SQL Server and ask the domain administrator to remove the user from those groups as well.

For example, let's say that you had granted the user MYCORP\Martha login access to SQL Server. However, in the Windows domain, Martha is a user of the MYCORP\SalesSupers global group, which has also been granted login access to SQL Server. As Martha moves out of the sales department and into marketing, she no longer needs access to the database. If you remove the login for MYCORP\Martha from the SQL Server, she can still gain access to SQL Server because at the domain level she is still a member of the MYCORP\SalesSupers global group. To fix the problem, ask the domain administrator to remove Martha from the SalesSupers global group. That makes sense, of course, because if Martha has moved to marketing, she is no longer a sales supervisor.

Q. I have a select group of individuals who need access to my SQL Server. I have created a global group of SQL users, placed the Windows accounts for the users who need access into that global group, and granted the global group login access to SQL Server. That worked great!

Now I am trying to make sure that nobody else can connect, no matter what, so I have denied the global group domain users login access. As soon as I did that nobody, including the users who could previously log in, can connect to SQL Server. Why?

A. First of all, understand that by default, only Windows administrators can connect to SQL Server. You have to give access to users or groups for them to gain access. You did that by creating the Windows global group, placing Windows users in it, and granting the global group login access to SQL Server. At that point, you should have stopped. Nobody else could connect because you never said they could.

By denying login access to SQL Server to domain users, you basically said that no way, no how, can anybody who is a member of the domain users global group ever connect to SQL Server, regardless of what any other login allows. Domain users are a very broad group that by default includes every user in the domain, including those users who were members of the global group to which you had granted login access.

To fix the problem, revoke the domain users login that you had denied. Revoking it essentially deletes it from SQL Server.

Q. When I am at work, I can connect to my SQL Server using Windows authentication. However, when I am at home, even though I can ping my server, I cannot use

Windows authentication to connect. Why?

A. At work, you are likely logging into a Windows domain that your SQL Server either belongs to or recognizes. For that reason, when you connect using Windows authentication, you have a valid set of credentials that SQL can pass to the Windows system with which to authenticate you.

When you connect from home, you most likely are not logging in to the domain. Even if you log in to your system at home, possibly even with the same username and password, the domain is different. If you present your home credentials to SQL Server with Windows authentication, it passes those credentials off to Windows, and Windows attempts to authenticate you. Since the Windows system doesn't recognize your home domain, or lack thereof, it cannot properly authenticate you, and your connection will fail. If you need to connect from home, and you do not log in to the corporate domain from home, you need to enable the SQL Server and Windows authentication mode discussed in this chapter, create a standard SQL login for you to use from home, and connect using SQL authentication.

Q. I log into my computer at work using a specific username and password. In SQL Server, I created a SQL login using the same username and password with which I log in to Windows. Now, when I try to connect to SQL Server using Windows authentication, I get an error message that reads "Login failed for user YOURDOM\Username."

A. Even though you have a standard SQL account that has the same username and password as your Windows login, they are two completely separate accounts that get authenticated by completely different mechanisms, and quite honestly, they have nothing to do with each other. They just coincidentally have the same name and password.

When you connect using Windows authentication, SQL passes your username and password to the Windows security subsystem. Windows then takes those credentials and authenticates you. In other words, Windows verifies that you are who you say you are. In your case, Windows is even able to validate you, it tells SQL Server that you are who you say you are, and it presents SQL Server with a list that contains your Windows user ID and the IDs of all the Windows groups to which you belong. SQL Server then looks in its list of logins for a Windows login that matches one of the IDs Windows gives it. It will not find one.

The login that you made was not a Windows login, it was a SQL login. It does not have the same ID as your Windows user, nor does it have the same ID as any of the groups to which you belong. As SQL Server scans the list of logins, it is looking for matching IDs, not matching names. It will not find a match, and you will not connect.

To fix the problem, do one of two things: Either connect using SQL authentication and resupply your username and password each time you connect, or delete the SQL login, create a new Windows login that references your existing Windows account, and connect using Windows authentication.

Chapter 6: Administration and Active Directory Integration

Introduction

Every organization creates data and has the need for a secure and reliable location to store that information. Although that statement is not news to anyone reading this book, for many small and medium-sized organizations, a secure and reliable location has been the key to the information technology puzzle. When Microsoft began development on the next release of SQL Server following version 6.5, one of its goals was to simplify administration. SQL Server 7.0 made tremendous headway in allowing smaller organizations to solve their data storage problems with SQL Server. Without dedicated database administrators, administering many relational database systems is out of the question. This problem was solved by providing features such as administration wizards to create objects and perform tasks such as adding users, backing up databases, and transferring data. SQL Server's autotuning and file autogrow features cut down on the amount of administration required to keep your database solution up and running. All of these features are complemented by the advanced techniques available to large organizations that have dedicated SQL Server experts. The result of this effort is a database system that can run a four-person small business all the way up to multinational organization with dozens of database administrators.

If your organization is like many, starting up your database server and verifying backups is not the end of your day. In addition to its dozens of enhancements and new features, SQL Server 2000 has extended its administration assistance with additional wizards, enhanced autotuning and integration with Windows 2000 Active Directory. Transferring databases and data has been simplified with the Database Copy Wizard and Data Transformation Services. Locating and using SQL Server in the enterprise is simplified with Active Directory. Administering your database server can be accomplished with graphical tools such as SQL Enterprise Manager with T-SQL commands or programmatically using SQL-DMO.

This chapter reviews these administrative tasks and the tools and techniques you can use to keep your database server and your organization reliable.

Windows 2000 Active Directory Integration

The Active Directory provides the hierarchical framework for security within the Windows 2000 network. Active Directory is a directory service that centralizes user accounts and security principal relationships. The security model used is *Kerberos* , an industry-standard authentication protocol. SQL Server 2000 integrates with Active Directory using the same Kerberos protocol.

Kerberos imparts mutual authentication services between client and server. The client's security credentials can be passed along, thus enabling the client to connect to multiple servers. In a method called *security account delegation* , the client authenticates to the first server using the Kerberos protocol. Then the first server impersonates the client for access to subsequent servers.

In order to use the security services integrated in the Active Directory, all SQL servers to which a client connects must be running on the Windows 2000 platform with Kerberos. Then you can further take advantage of Active Directory integration through configuration of the SQL servers, databases, and publications.

About Active Directory

As we've stated, Active Directory for Windows 2000 is a *directory service* , which is a database that organizes network accounts and resources. A directory service is often used as an index for finding resources on the network—somewhat similar to how people use the White Pages to look up telephone numbers. Directory services take this concept a step further by providing a relationship between users and resources such that security and policies can be applied. *Policies* are rules that govern the way that resources and accounts interact. Policy-based networking is a powerful approach to managing a network; many functions can be automated through policies, and the time and effort involved in administration is thus greatly reduced.

Active Directory is extensible. In other words, new objects (that represent resources) can be added to the database, or new attributes for existing objects can be added. In fact, Active Directory is based on an Extensible Storage Engine (ESE) database, the same as is used for Microsoft Exchange Server. Applications can extend the schema, and consequently the functionality, of Active Directory.

The directory service is a multimaster database that organizes objects representing user accounts, group accounts, and network resources in a hierarchy established by an administrator. *Multimaster* is a new concept for this next generation of Windows NT. In the legacy NT domain structure, a single primary domain controller (PDC) could own the additions and changes to the database. That PDC was a single point of failure and a traffic bottleneck. In Windows 2000, each domain controller (DC) shares ownership of the Active Directory database. A change can be made on any DC and then propagated to the rest of the network through replication.

Active Directory domains use the *Domain Name System (DNS)* as the naming system for domains. These domains are organized into a hierarchy, with child domains and grandchild domains, and so on. Within each domain are containers that can be structured into a tree formation. These containers are called *organizational units (OUs).* OUs contain objects that represent user accounts and network resources.

Group policies are an Intellimirror feature whereby a user can move from PC to PC and always receive the same environment. Group policies do far more than that, however. They provide the foundation for policy-based networking. Group policies are applied to Active Directory OUs, domains, and sites. An object that exists within an OU uses the group policy of each of the following: the local computer policy, the site group policy, the domain group policy, and each group policy of the OUs from the root of the domain to the OU containing the object.

Active Directory might look similar to other directory services. This apparent similarity is due to the fact that it is based on a directory service standard, X.500. X.500 describes a specific naming and hierarchical organization and structure for a directory service. Whereas Active Directory implements many X.500 elements, it also implements some unique ones as well. In addition, Active Directory uses *Lightweight Directory Access Protocol (LDAP),* a protocol that enables clients to access a directory service. Since this is an open standard, an LDAP client can access and utilize the Active Directory, which means that you are not dependent on a proprietary client or protocol.

Registering SQL Servers in Active Directory

To begin integrating SQL Server 2000 with Windows 2000 Active Directory, you need to register the SQL servers. This will add a server object to the Active Directory, which you can view in the Active Directory Users and Computers.

SQL Server Properties

Registering the SQL Server is the first step toward Active Directory integration. The result of this action is that database properties are stored in Active Directory. These properties include the database's description, its size—even information about the last backup. Once you have this information stored in Active Directory, it is much easier to manage multiple SQL Server databases.

In order to register a SQL Server object, you need to access the SQL Server properties. To do this:

1. Log on with an account that has administrator privileges.
2. Click Start | Programs | Microsoft SQL Server | Enterprise Manager.

3. In the window, navigate to the SQL Server group that contains the SQL Server you are adding.
4. Expand the group so that you can see the SQL Servers.
5. Right-click your server and select Properties.
6. Next, click the Active Directory tab, which is shown in Figure 6.1 .



Figure 6.1: The Active Directory dialog box in SQL Server Properties.

7. Click the Add button.
8. You will know that the action was successful when you receive the dialog box shown in Figure 6.2 . Click OK to close.

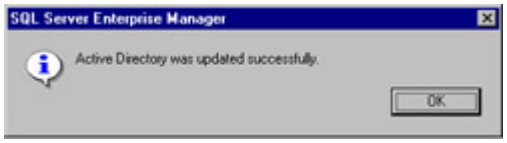


Figure 6.2: Successful update of Active Directory properties.

After you add the SQL Server object, your next step is to update the attributes of the SQL Server object by clicking the Refresh button on the Active Directory tab. Then you can see the SQL Server object appear in the Active Directory Users and Computers. To open this console, click Start | Programs | Administrative Tools | Active Directory Users and Computers. You need to click the View menu and select the Advanced Properties option, if you have not already done so. The SQL Server information you will see is exhibited in Figure 6.3 .

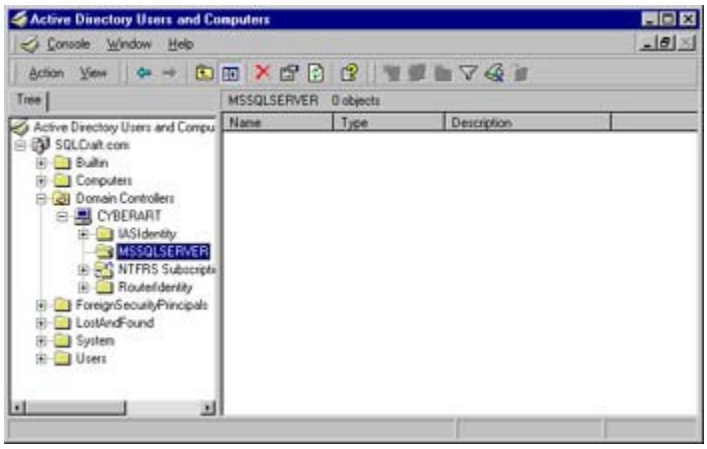


Figure 6.3: SQL Server information appearing in Active Directory Users and Computers.

Tip If you want to remove the SQL Server from the Active Directory, you can do so. However, you will not only remove the SQL Server; you will also remove all its databases and publications. Removing these elements might be necessary if you have major Active Directory domain changes. To remove the server, right-click SQL Server in the Enterprise Manager, and select Properties. Click the Active Directory tab. Click the Remove button.

sp_ActiveDirectory_SCP

Enterprise Manager is not the only way to add a SQL Server object to (or remove a SQL Server from) Active Directory. You can also use Transact-SQL. The process begins by executing sp_ActiveDirectory_SCP. To add the SQL Server, set @action='CREATE'. To remove the SQL Server, set @action='DELETE'.

Registering Databases in Active Directory

To gain the most from Active Directory, you should register individual databases as well as the SQL servers. Applications can then connect to a database by searching for the registered information in Active Directory. This enables an administrator to change the name or the location of the database, without being required to change the application. Active Directory becomes the locator service on behalf of the client.

Database Properties

The properties of each database in Enterprise Manager provide the means for listing that database in the Active Directory. To do so:

1. Log on with an account that has administrator privileges.
2. Click Start | Programs | Microsoft SQL Server | Enterprise Manager.
3. In the window, navigate to the SQL Server group that contains the SQL Server you are adding.
4. Expand the group so that you can see the SQL Servers.
5. Expand the SQL Server that contains your database.

6. Right-click the database.
7. Select Properties from the pop-up menu.
8. Click the Options tab.
9. Check the box to list this database in Active Directory, as shown in Figure 6.4 .

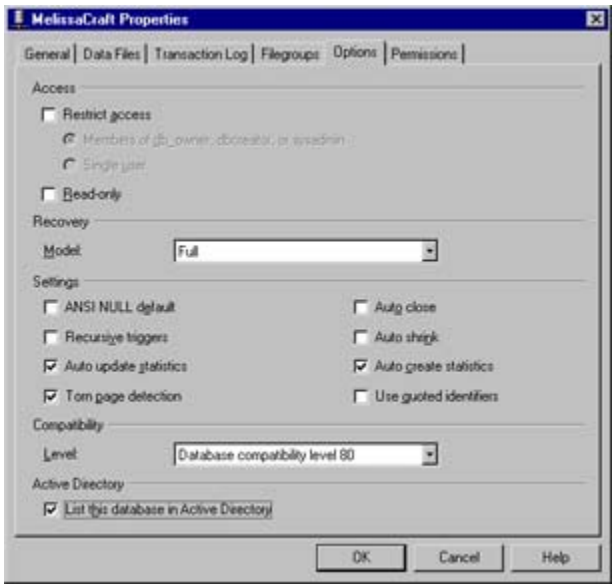


Figure 6.4: Listing a database in Active Directory.

Once you've listed the database, you can look at the Active Directory Users and Computers under your SQL Server object to see the new database object. This is shown in Figure 6.5 .

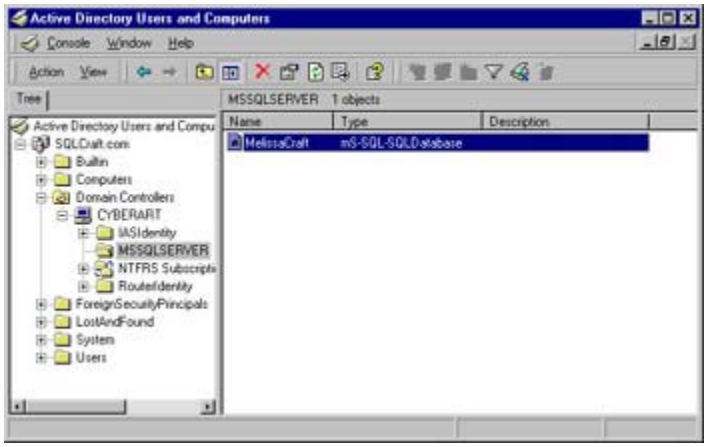


Figure 6.5: New database object shown in Active Directory.

SQL Replication Services and Active Directory

When you configure SQL replication services, you could confuse them with Active Directory replication. Before you configure SQL replication, you should understand how it is different from Active Directory replication. You should also understand that, although SQL Servers replicate data on their schedule, the information that is listed in

Active Directory will synchronize based on the Active Directory replication schedule. That means that you could change some aspect of the SQL Server and, though it might be updated within the SQL Server replication topology, it will not change within the entire Active Directory until the Active Directory replication is completed on its own schedule. Although this might happen rarely, you need to know these replication differences if conflicting information between SQL and Active Directory becomes an issue in your environment.

SQL Server 2000 Replication

SQL Server 2000 replication accomplishes the synchronization of data among multiple copies of a SQL Server database or of copying the data from a table to another table located on a separate server. Through SQL Server's native replication, you can design a distributed database system among multiple physical servers. The replication occurs automatically, with no human intervention required. One of the enhancements to SQL Server 2000's replication component is the ability to replicate schema updates. There is no need to recreate publications and subscriptions whenever the schema is changed.

Tools and Techniques for Examining SQL Server Objects in Active Directory

Almost all of your administration will be executed in the SQL Server Enterprise Manager console. Even when the objects appear in the Active Directory Users and Computers console, you will not do much administration in this console. For one thing, the SQL Server object and database object appear without extensive attributes that you would find in the SQL Server Enterprise Manager. That doesn't mean that the attributes are not listed in Active Directory, simply that you would need to use a different method to adjust them. Obviously, the easiest route is to use Enterprise Manager, but let's explore another option: Lightweight Directory Access Protocol, or LDAP.

LDAP can be used to perform operations on Active Directory, to search it and to integrate with applications. One of the tools that is available on the Windows 2000 CD-ROM in the Support\Tools directory is LDP. LDP can add, change, or remove objects and their properties from Active Directory. Not only that, but LDP works with any LDAP-compatible directory service. LDP tends to be a complex tool, and its discussion is far beyond the scope of this book. However, you can look into LDP or other LDAP tools for managing the SQL Server and database objects and properties in Active Directory. Figure 6.6 shows the LDP tool screen.

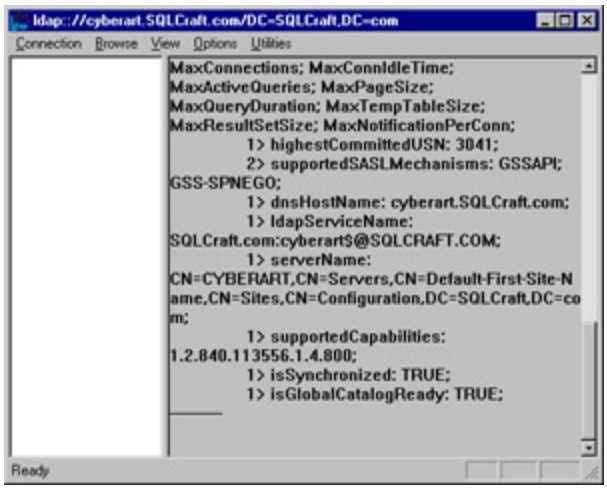


Figure 6.6: The LDP tool.

A second tool that you might consider using is ADSI Edit, another utility found in the Support\Tools directory on the Windows 2000 CD-ROM. ADSI stands for Active Directory Service Interfaces. ADSI Edit is another method of accessing Active Directory. The ADSI Edit tool is somewhat easier to use than the LDP tool due to its interface.

When you use ADSI Edit, you should consider the display specifiers for the objects that you are viewing. Display specifiers determine what an administrator or even a user sees when looking at any particular object. Figure 6.7 illustrates the ADSI Edit screen.

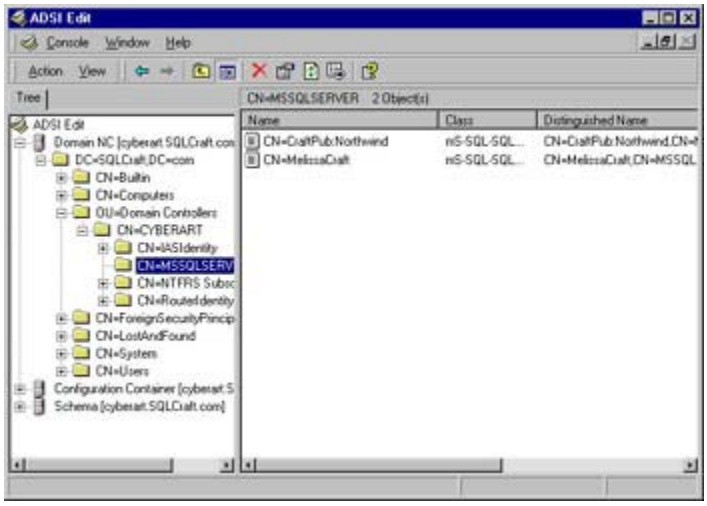


Figure 6.7: The ADSI Edit tool.

In order to modify SQL Server properties in ADSI Edit, you may follow this procedure after the support tool has been installed:

1. Log on with a user account that has administrative privileges.
2. Click Start | Programs | Windows 2000 Support Tools | Tools | ADSI Edit.

3. Open the domain node where the SQL Server resides. It will be named DC=sub,DC=domain,DC=com, which represents a DNS name of sub.domain.com.
4. Look within the OU or container that holds the SQL Server object. It will be listed as CN=SERVERNAME.
5. Expand the SQL Server object so that you can see the CN=MSSQLSERVER object below it.
6. When you click the CN=MSSQLSERVER object, you will see within the right-hand window pane any databases that were listed as Active Directory objects.
7. To change a SQL Server property, right-click the CN=MSSQLSERVER object in the left-hand pane or on the selected database in the right-hand pane and select Properties from the pop-up menu.
8. The properties dialog box is displayed in Figure 6.8 . Click the arrow in the "Select a property to view" drop-down box to view the list.

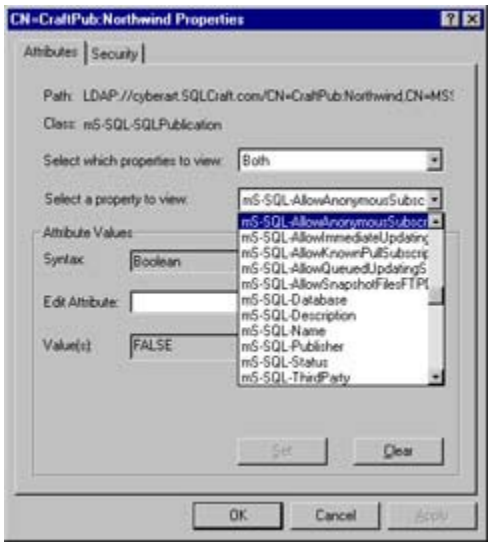


Figure 6.8: Editing attributes in ADSI Edit.

9. Scroll down to the properties that begin with *ms-SQL-* . These are the SQL Server properties, as you might have guessed.
10. In the Attribute Values section, you will be able to view the Syntax of the attribute—whether it is a Boolean, String, Integer or other type of attribute—as well as the current value assigned to that attribute listed in the Value(s) box. These are grayed out. To change the value of the attribute, click the Edit Attribute box and type in the new value you want to be assigned to it.
11. Click Set to apply the new value, and it will appear in the Value(s) box.
12. When the process is complete, click OK to exit the dialog box.

As a .NET Enterprise Server, SQL Server's replication enables a Web site to maintain multiple copies of a database in separate locations. This capability supplies a level of

redundancy, should one of the servers suffer a failure. It also supplies a measure of scalability if the number of users for the database expands beyond the capabilities of the existing server(s) supporting it.

Being able to maintain live copies of databases is a captivating topic when you consider the applications that such a capability can have on the Internet. Multiple servers can provide responses to a vast number of users. Administrators can scale the database services as needed, rather than take the trouble to upgrade a database server box with increasingly superior hardware. Knowledge bases, electronic mail, knowledge management, data warehousing, and enterprise resource planning systems—all built with database technology—readily gain from a solid replication system.

Active Directory Replication

Active Directory replication is a completely different system from that of the legacy Windows NT domain replication. In Active Directory, each DC holds various partitions of Active Directory.

Replication in the Windows 2000 system occurs between all the DCs in a multimaster system. An administrator can execute a change on any DC. Then the change is synchronized among the DCs that carry that same partition. For example, if an administrator makes a change to the Schema, that change is replicated to all the other DCs in the forest. However, when an administrator makes a change to a domain object, the change is replicated to all DCs that are members of that domain. If that object change has attributes that are copied to the GC, all the Global Catalog servers across the entire forest will replicate that change as well.

One advantage of multimaster replication is the redundancy provided by multiple copies of the various partitions. Another advantage is the ability to make changes to Active Directory locally, so that an end user can change a password on his or her local DC without needing to locate a single server that could possibly exist across a wide area network (WAN) link. This very issue of single point of change existed with Windows NT, so this is a major improvement. Multimaster systems also carry a disadvantage: conflicts. It is very possible that an administrator can make a change on one particular DC while another administrator makes a different change to the same attribute on a different DC. When the DCs replicate their changes, the conflict must be resolved. The current system within Active Directory uses an algorithm to determine which change was made last, and that change wins.

There are two types of Active Directory replication: intrasite replication and intersite replication. *Intrasite replication* is the process of synchronizing DCs that exist within the same site. A *site* is a collection of well-connected IP subnets. The term *well-connected* refers to the fact that these IP subnets are all linked by highly reliable and available network connections. This description usually refers to local area network (LAN) connections between the IP subnets, but should you want to include a slow WAN link,

you can lie about its well-connected status and include it in a site. Active Directory won't make a big fuss about it, but you might suffer some resulting performance issues for logons and Active Directory queries. Luckily, you can change your site configuration at any point, so you might experiment until you reach an optimal configuration.

Replication traffic within a site (intrasite replication) is fast over LAN links because it is not compressed and it runs every 5 minutes between the DCs in the site. Replication traffic among sites (*intersite replication*) can be managed to perform well over WAN links because it is compressed and runs only at the times that the administrator specifies.

The *Knowledge Consistency Checker (KCC)* manages intrasite replication topology in Active Directory. The KCC automatically configures the replication topology among DCs within the site. The KCC configures a system whereby DCs are linked in a circle with no more than three hops between the DCs. With more than four DCs in a site, the KCC configures multiple interlinked circles.

Connection objects in the Active Directory Sites and Services console represent each intrasite link. The KCC will generate these automatically, or an administrator can add connection objects as needed.

Intersite replication is the process of synchronizing the changes made on DCs in one site with the DCs in other sites, and vice versa. The intersite replication topology must be fully configured by the administrator. Once two separate sites are defined, they will not communicate until they are linked by a Site Link object.

To establish intersite replication, the administrator must first define the sites. This is done by opening the Active Directory Sites and Services console, right-clicking the Sites node, and selecting New Site from the pop-up menu. The new site will need to be connected to a site link, and you can use the DefaultIPSiteLink until you define your own site links. Then you should assign IP subnets to each site, so that DCs, member servers, and client computers will be able to detect the site to which they belong. You do this by navigating below the Sites node to the Subnets container, right-clicking the container, and selecting New Subnet. In the resulting dialog box, type in the subnet and mask, then select a site to which to add the new IP Subnet object. After assigning subnets, the administrator creates Site Links between those sites that are directly connected by a network connection. For those sites that are not connected directly to each other, the administrator creates a site link bridge between site links that share a common site. Management of the traffic occurs when the administrator selects bridgehead servers for traffic to travel through from one site to another. The administrator then creates connection objects because the KCC will not create them automatically between sites.

Once you install a SQL Server and list it in Active Directory, you see no changes in the Active Directory Sites and Services console. This console is shown in Figure 6.9 .

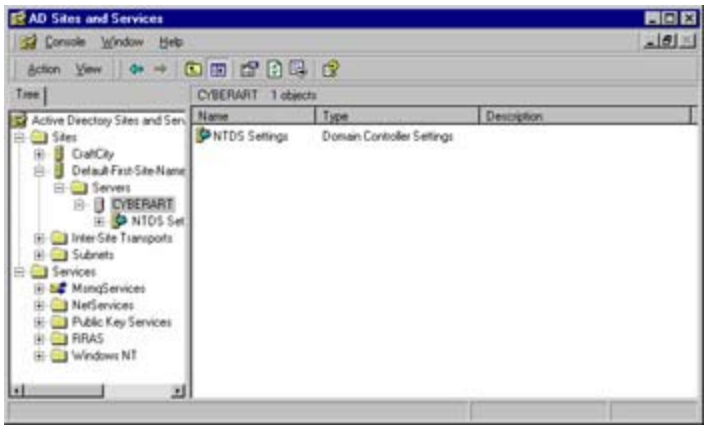


Figure 6.9: SQL Server changes shown in the Active Directory Sites and Services console.

Replication, under SQL Server, is an architecture of publishing and subscribing that is extended with distribution. With replication, a SQL Server database:

- Is scalable
- Has higher performance
- Is promptly available
- Is more reliable due to redundancy

Tip For more information about replication, see Chapter 12 , "Database Replication Techniques and Configuration."

Registering Publications in Active Directory

Once you register the publication in Active Directory, applications and users can use Active Directory to query for publications. Queries can be performed based on properties, so a user does not need to know the SQL Server instance or the database name. Instead, the user can browse for the publication that meets his or her criteria. Once a publication is found, and if a user is allowed, he or she can subscribe to the publication.

Because the replication of the Active Directory properties does not update on the same schedule as the SQL Server replication, there could be a discrepancy between the two listings. When you are trying to discern which value is correct, you should select the value listed in SQL Server Enterprise Manager.

To register a publication in Active Directory, you must first list the SQL Server object. The process for registering the publication in Active Directory is as follows:

1. Log on to the server with an administrative account.
2. Click Start | Programs | Microsoft SQL Server | Enterprise Manager.
3. Navigate to the publication.

4. Right-click the publication.
5. Select Properties from the pop-up menu.
6. Make certain that the General tab is selected.
7. Check the box to list this publication in Active Directory, as shown in Figure 6.10 .

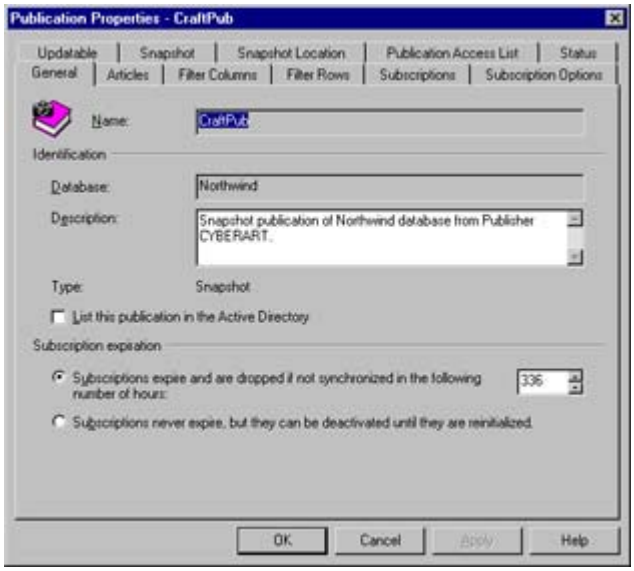


Figure 6.10: Listing a publication in Active Directory.

You can also list a publication in the Active Directory at the time that you run the Create Publication Wizard. You can check the check box in the Select Publication Name and Description dialog box to list this publication in Active Directory.

When creating new publications with Transact-SQL, you can use the T-SQL statements sp_addpublication and set @add_to_active_directory='TRUE'. Or you can use sp_addmergepublication (for merge replication publications) and set @add_to_active_directory='TRUE'. With existing publications, you can use sp_changepublication or sp_changemergepublication and set @property= publish_to_ActiveDirectory, @value='TRUE'.

Locating and Using Publications in Active Directory

The Pull Subscription Wizard is one of the ways in which to browse or subscribe to a publication that has been listed in Active Directory. To use this method:

1. Log on at the client.
2. Click Start | Programs | Microsoft SQL Server | Enterprise Manager.
3. Select the SQL Server by navigating to it and clicking it in the left pane.
4. Click the Tools menu.
5. Select Replication.
6. Select Pull Subscriptions to SERVERNAME.

7. Click the button for Pull New Subscription. Alternatively, you can reach the Pull Subscription Wizard by selecting the Tools menu, then selecting the Wizards option and, below the Replication heading, selecting the Pull Subscription Wizard.
8. Check the box for "Show advanced options in this wizard" if you want to see all the options.
9. Click Next.
10. Select the radio button for "Look at publications in the Active Directory or specify publication information … ." This dialog box is shown in Figure 6.11 .



Figure 6.11: Looking into Active Directory.

11. Click Next.
12. To browse Active Directory, you can click the Browse button on the Specify Publication dialog box, as illustrated in Figure 6.12 .



Figure 6.12: Specify Publication.

13. Clicking Browse leads to the Find SQL Server Publications search box. On the SQL Server Publications tab, you can type in a name for the publication you are

looking for or leave it blank. Click the Advanced tab, which is shown in Figure 6.13 .



Figure 6.13: Advanced search options for the Active Directory listed publications.

14. You may click the Field button and select the attributes of the Server you would like to search for. When you are finished, click the Find Now button.
15. Select a publication from the list by double-clicking it, and the dialog box will automatically return to the Pull Subscription Wizard.
16. Click Next.
17. Select a database in which to create the subscription, and click Next.
18. Select whether the SQL Server needs to initialize the schema and data, and click Next.
19. Select where to pull snapshot files from, and click Next.
20. Select a schedule for the Distribution Agent, and click Next.
21. On the final screen of the Wizard, review the pull subscription options, and click the Finish button.

Analysis Services and Active Directory

In SQL Server 7.0, analysis services were called *OLAP services* . OLAP stands for *online analytical processing* . Generally, *analysis services* are tools provided to assist in developing and maintaining data used for business analysis and decision making. The Analysis Service server forms cubes of data (aggregate data) for use in multidimensional analysis. *Multidimensional analysis* is a system of multiple queries that scrutinize data from different perspectives, or *dimensions* . Each query can look at a different aspect of the data, but all queries can be constructed and processed against the same data set.

Registering Analysis Servers in Active Directory

In order to allow people to search Active Directory to find Analysis Servers, you must register them in Active Directory. The following procedure accomplishes this task:

1. Log on as a user with administrative privileges.
2. Click Start | Programs | Microsoft SQL Server | Analysis Services | Analysis Manager.
3. Right-click the SQL Server below the Analysis Servers container.

4. Select Properties from the pop-up menu.
5. Click the Active Directory tab.
6. Check the box for Enable Active Directory Registration, as displayed in Figure 6.14 .

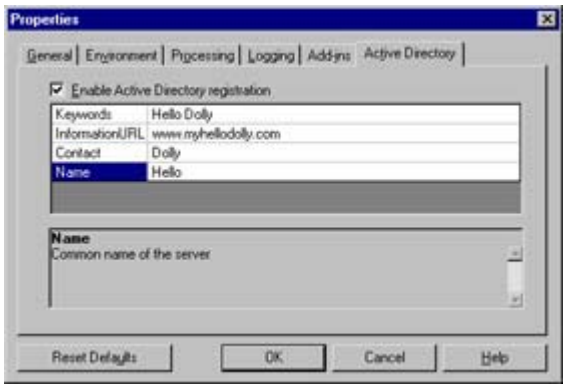


Figure 6.14: Registering Active Directory.

7. In the right-hand box next to Keywords, type in any keywords that will describe the analysis server.
8. In the right-hand box next to InformationURL, type in a Universal Resource Locator (URL) that leads to a Web page providing information about the server.
9. In the right-hand box next to Contact, type the name of the contact for the server.
10. In the right-hand box next to Name, type in a commonly used name for the server.
11. Click OK to complete the process.

Tools and Techniques for SQL Server Administration

Database management encompasses more than creating a database and backing it up on a periodic basis. In fact, database administrators find themselves with a long list of responsibilities:

- SQL Server installation
- Analysis Server installation
- Client installation
- Security configuration
- Database configuration
- Permissions assignment
- Configuration for redundancy and fail-over
- Creation of publications
- Subscription creation
- Replication monitoring and management
- Integration with outside services

Database administrators perform these functions and much more. We'll look at a few of the utilities that they use in the course of a day.

Windows 2000 Active Directory

Other than ADSI Edit and LDP, which were discussed in a previous section of this chapter, the main tool that a database administrator might use is the Active Directory Users and Computers console. This is a standard console available on any Windows 2000 domain controller or on a computer on which Active Directory administrative tools are installed.

Active Directory is a new feature for Windows 2000 Server, which is built on top of NT architecture. Therefore, it is a version 1.0 release of a directory service. SQL Server 2000 followed Windows 2000 in its release to the market. SQL Server does integrate with Active Directory, with the ability of storing details about the SQL Server objects, but it is nonetheless a version 1.0 integration. The potential of Active Directory integration is phenomenal. Data mining and reporting are two areas that will eventually benefit from Active Directory/SQL Server integration, since both are activities that benefit from being able to locate and utilize information from multiple data repositories.

Active Directory Users and Computers

When you configure a server to have separate SQL Server accounts, you will create users in the SQL Server Enterprise Manager. However, when you configure a SQL Server to integrate with Windows 2000 security, you will create user accounts through the Active Directory Users and Computers console. Not only that, but you will use the console to view SQL Server objects. Some organizations maintain a group of network administrators who are assigned the task of creating and managing users, groups, and

other objects found in Active Directory and a second group of database administrators (DBAs) who manage the databases running across the enterprise. Others, however, assign these tasks to the same group. If you are assigned the role of both network administrator and DBA, you need to create the user accounts first, then assign database permissions.

You can start the Active Directory Users and Computers console by clicking Start | Programs | Administrative Tools | Active Directory Users and Computers. This console is shown in Figure 6.15 .

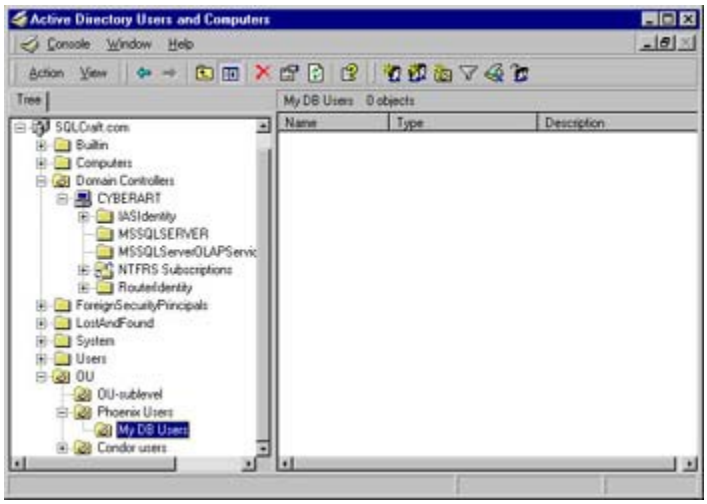


Figure 6.15: The Active Directory Users and Computers console.

When you need to add a user to Active Directory, you use this console. User accounts can be located within any OU) the standard container within Active Directory. An administrator can construct as many OUs as necessary in a hierarchical structure. To create a user account:

1. Open the Active Directory Users and Computers console.
2. Navigate to the OU where the new user account will be located.
3. Right-click the OU.
4. Select New.
5. Select User. You will see the dialog box shown in Figure 6.16 .

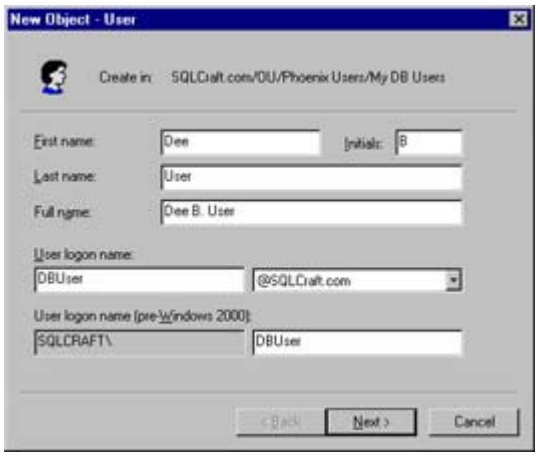


Figure 6.16: The user creation dialog box.

6. Complete the user information, and click Next.
7. Complete the password information in the next dialog box, illustrated in Figure 6.17 .

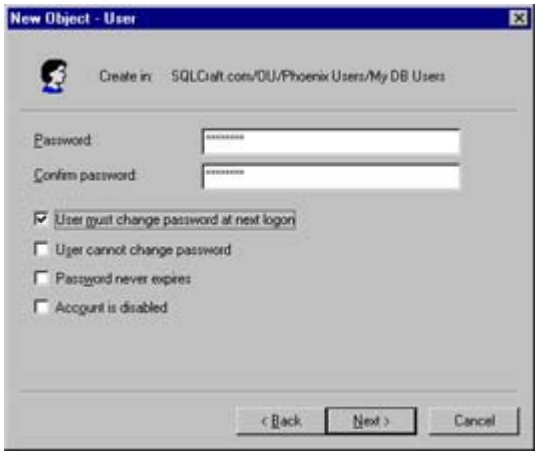


Figure 6.17: Password information dialog box.

8. Click Next.
9. Review the information, and click Finish.

This process creates a user account with only minimal information. In order to complete more detailed information, right-click the user account you created, which will now appear in the right-hand pane of the Active Directory Users and Computers console, and select Properties from the pop-up menu. The user account properties dialog box will appear. An example of this dialog box is shown in Figure 6.18 . As you can see, you can configure many details (requiring multiple tabs!) for a user account.



Figure 6.18: The User Properties dialog box.

If you are a DBA and not a network administrator, you will find that your interest in user accounts lies in granting them some measure of access to your SQL Servers. Granting SQL Server access is configured in the Enterprise Manager.

Microsoft Management Console

One of the improvements in Windows 2000 was a common framework for administrative consoles. This framework manifested into the Microsoft Management Console (MMC). MMC is a central application used to manage any and all facets of the Windows 2000 operating system.

One of the more confusing aspects of running legacy Windows NT servers was the inconsistency of the administrative tools. Some tools were “right-clickable”; others didn’t support pop-up menus. Some tools used an organizational tree structure, others used icons, and so on. MMC was created to streamline and simplify daily management of Windows 2000 Server systems.

The MMC itself is a shell. It hosts other applications and utilities. This makes the MMC extensible. Not only will future development of Microsoft BackOffice products use this shell, but administrators can develop their own unique consoles to help organize their everyday tasks. An MMC can be created and saved as a file with an .MSC extension. Once a console has been saved as a file, an administrator can distribute that console to users, groups—even computers.

The MMC is extensible with snap-ins, additional utilities that work within the MMC shell. To customize the MMC:

1. Click Start | Run.

2. Type **MMC** and press Enter.
3. Click the Console menu.
4. Select Add/Remove Snap-In.
5. Click the Add button.
6. Select the snap-in from the list, and click Add.
7. Select the computer that this snap-in will manage, and click Finish.
8. Continue to add snap-ins until all that are required are added, then click the Close button.
9. Click Close when the process is complete.

Tip To reduce the time and effort it takes to open and close multiple console applications as you move from task to task, you can create a custom MMC that includes all the consoles that you commonly use. For example, you might want to add the SQL Enterprise Manager snap-in, the Analysis Manager, and Active Directory Users and Computers to a single custom MMC.

The Delegation of Administration Wizard

In smaller organizations, many DBAs are given the task of managing a set of users for all their network needs, in addition to handling database access. These DBAs need to create new users, assign permissions, update the user accounts, and manage their general network use.In the old Windows NT domain world, there was no choice but to make these DBAs part of the Domain Admins group, thus granting the database administrators many more rights than were necessary for performing their jobs.

In Windows 2000, this is no longer an issue. Active Directory enables an administrator to delegate rights to an OU. For DBAs with extended job functions, this means that they can be granted their own OUs with their own user accounts and the right to manage those user accounts. This feature can be executed with the Delegation of Administration Wizard as follows:

1. To start the wizard, navigate to the OU that will be delegated to the DBA, and right-click it.
2. Select Delegate Control from the pop-up menu.
3. Click Next at the Welcome screen for the Delegation of Administration Wizard.
4. The next screen will ask you for the user or group account to whom you want to give control. Click the Add button, and add the user(s) and/or group(s) to the list by selecting the accounts and clicking the Add button for each, then clicking OK to close the dialog box.
5. Whenever possible, delegate control to a group and add the users to it, even if there is only a single user. This method makes it easier to adjust your administrative team later. Click Next.
6. The next dialog box allows you to select the administrative functions that will be given to the accounts you selected in the previous dialog box. The most common administrative functions are listed with check boxes at the top, or you can click

the radio button to customize the abilities this administrative group will have. When you're finished with your selections, click Next.
7. Finally, review the information and verify that it is correct, then click the Finish button.

SQL Server Enterprise Manager

SQL Server Enterprise Manager is the main administrative tool for managing a SQL Server. Enterprise Manager is a complete SQL Server management tool based on SQL-DMO. With it you can:

- Start and stop SQL Server.
- Assign the system administrator's password.
- Schedule jobs.
- Manage SQL Server users and security.
- Configure servers.
- Register servers, databases, and publications in Active Directory.
- Monitor the server with alerts and e-mail notification.
- Manage all aspects of a SQL database.

Enterprise Manager organizes servers into server groups for simplification of administration. Using these groups, an administrator can limit access for users to those items within a particular server group. The server groups can be used for applying commands across several servers at once, instead of a single server at a time. SQL Servers are automatically placed within a default server group named SQL SERVER GROUP. To make your own:

1. Log on as a user with administrative privileges.
2. Click Start | Programs | Microsoft SQL Server | Enterprise Manager.
3. Right-click the Microsoft SQL Servers container.
4. Select New SQL Server Group from the pop-up menu. The Create New Server Group dialog box shown in Figure 6.19 will appear.

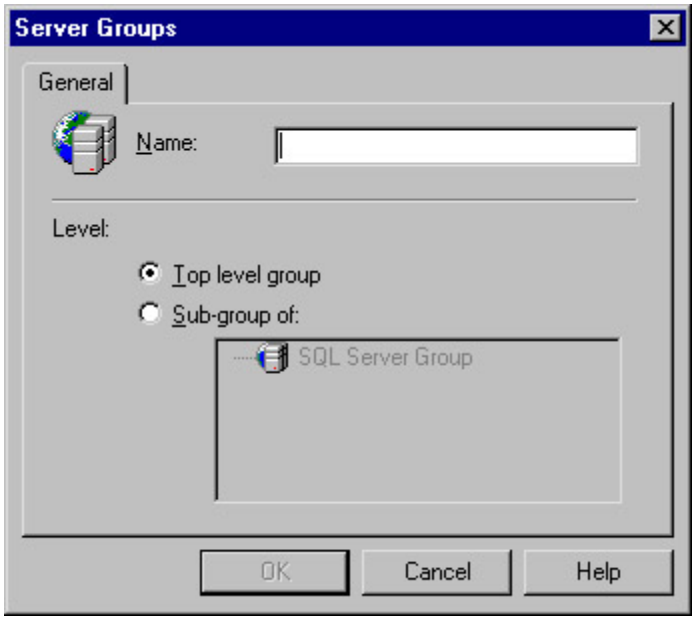


Figure 6.19: Creating a new server group.

5. Type a name for the new server group. You also have the option of placing this new server group into a hierarchy by making it a subgroup of another server group. Otherwise, it should be a top-level group.
6. Click OK.

Once the server group is created, you can populate it with servers. To do so, right-click a server group, and select New SQL Server Registration from the pop-up menu. Click the Next button at the Welcome screen. Select a SQL Server from the available servers on your network. Click the Add button to move each server into the Added Servers list, then click Next. On the following screen, select the security type, and click Next. Select or create a server group, and click Next. Review the information on the final screen, and click the Finish button to add your server to the selected group. Click Close in the message box stating that your registration is successful.

At this point, you can begin managing user accounts from within Enterprise Manager. To grant a user account access to a SQL server:

1. Log on as a user with administrative privileges.
2. Click Start | Programs | Microsoft SQL Server | Enterprise Manager.
3. Navigate to the SQL Server you are administering, and expand the Security container.
4. Open the Logins container.
5. Right-click the Logins container.
6. Select New Login from the pop-up menu.
7. If the Authentication type is correct (Windows Authentication), click the ellipsis (…) button next to the name box to select a user or group account from Active Directory. Otherwise, you will be limited to accounts within SQL Server only.
8. Select the account, click the Add button, and click OK to finish that dialog box.

9. You will be returned to the SQL Server Login Properties box, which is shown in Figure 6.20 .

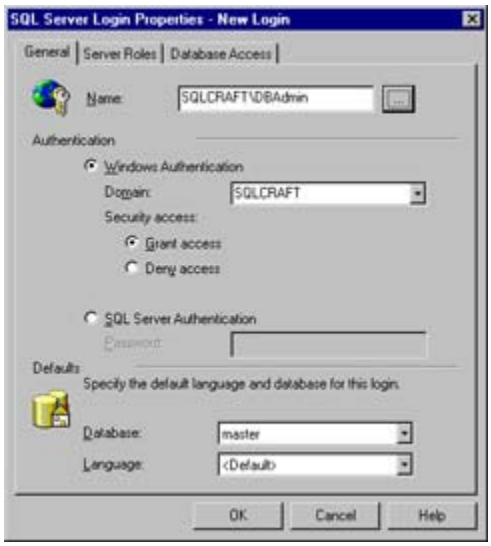


Figure 6.20: Login properties.

10. Make certain that the database and language are correct in the dialog box, or leave them as the defaults, then click the Server Roles tab.
11. Select the role or roles that this user account will be granted by checking the appropriate boxes.
12. Click the Database Access tab.
13. Select the databases to which you are granting access with the role you selected.
14. In the lower box of the window, select the type of role to grant to that particular database by checking the appropriate boxes. (Public is the default.)
15. Click OK to complete the login creation process.

Logins are one method of granting Active Directory user accounts access to SQL Server databases. Within each database, you can create user access that prompts new logins to be created. All you need to do is:

1. Open Enterprise Manager.
2. Navigate to your selected database.
3. Expand the database.
4. Right-click the Users container.
5. Select New Database User from the pop-up menu.
6. Use the drop-down box to select <new>, which will prompt the Login Properties box to open and create a new login.
7. When you have completed the new login, you will see the login name listed in the drop-down list of the New User properties box, which is shown in Figure 6.21 .



Figure 6.21: New user creation.

8. You can change the username if you want, and select the database role permissions.
9. Click OK to complete the new user creation process.

Enterprise Manager is also useful for administering component services of SQL Server. These services are:

- Microsoft Distributed Transaction Coordinator (DTC)
- Microsoft Search
- SQL Server Agent

You are enabled to only start or stop the Microsoft Distributed Transaction Coordinator service within Enterprise Manager. To do so, select the SQL Server and expand its container. Navigate to the Support Services container and expand that. Right-click Distributed Transaction Coordinator, and select Stop (or Start, if the service has already been stopped) from the pop-up menu.

Microsoft search is located in the Support Services container as well as the DTC. The feature is called Full-Text Search. You are granted more options with managing search: You can start and stop the service, clean up catalogs, and view the Microsoft search properties, which are shown in Figure 6.22 . All these options are available on the pop-up menu when you right-click the Full-Text Search icon.

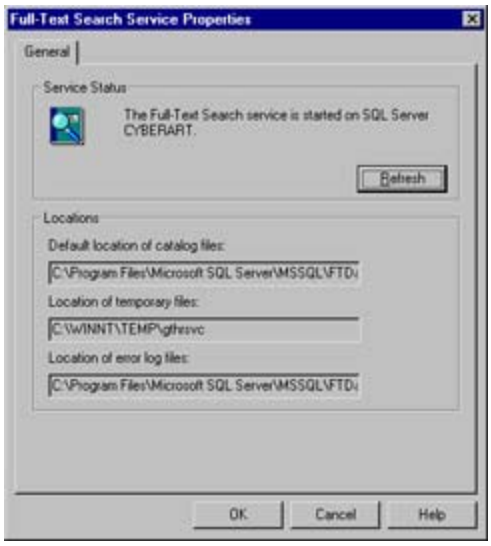


Figure 6.22: The Microsoft Full-Text Search Service properties dialog box.

The SQL Server Agent service is located within a different container in Enterprise Manager—the Management container located below your selected SQL Server. The SQL Server Agent contains three other objects: Alerts, Operators, and Jobs. To manage the agent itself, right-click it, and select Start or Stop to change the service's status. You can also select Display Error Log or explore the options below the Multi-Server Administration option, in which you can make the server a master or a target; create a new alert, operator, or job; or open the SQL Server Agent's Properties dialog box, which is shown in Figure 6.23 .

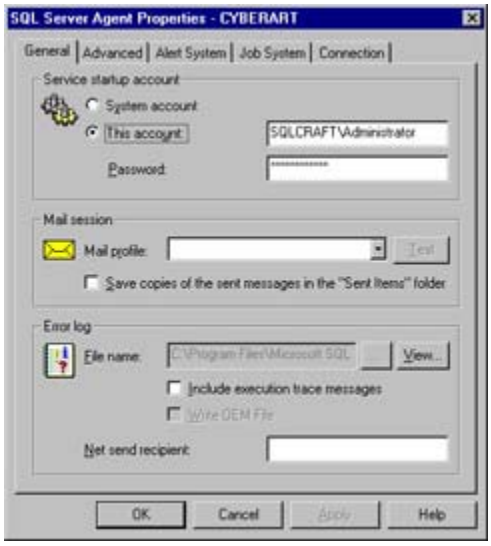


Figure 6.23: SQL Server Agent Properties.

Several tabs are available in the SQL Server Agent Properties dialog box. Table 6.1 details the options within each tab.

Table 6.1: SQL Server Agent Options

Tab	Options
General	Select the SQL Server Agent service startup account, establish the mail profile, and select or view the error log.
Advanced	Set the restart options for the service, configure event forwarding, set the CPU Idle settings to optimize performance.
Alert System	Format addressing of pager e-mails, establish a fail-safe operator.
Job System	Adjust job history log settings, set job execution parameters, select a proxy account.
Connection	Select the Authentication type (Windows or SQL Server) and set login timeout, view SQL Server alias.

One of the other things that Enterprise Manager provides is a method for monitoring your replication between publishers and subscribers. You can also manage the agents that are involved in the various types of replication. To view these options, navigate to the SQL Server you have selected. Open the Replication Monitor folder. You can view publishers and right-click on each to manage them. By right-clicking on an agent listed below a publisher, you can push new subscriptions, reinitialize all of them, view the agent's properties, and refresh the settings.

SQL Server MMC Snap-Ins

Aside from SQL Server Enterprise Manager, three other MMC snap-ins can help with managing the SQL Server environment:

- Component Services
- Meta Data Services
- SQL Analysis Services

Component Services comprises separate utilities that manage services from the same MMC. Of primary importance to the SQL database administrator, this console provides access to the Microsoft *Distributed Transaction Coordinator (DTC)* . DTC manages distributed transactions. A *distributed transaction* uses data from multiple databases, whether they are located on the same system or on separate servers. The DTC contains two options: a list of transactions and statistics. The Transaction Statistics option is shown in Figure 6.24 . Both of these options can provide valuable information to a DBA. If an application involves a transaction across multiple data sources, coordinating it can ensure that individual transactions that fail will not skew the result. The DTC Service provides coordination against inconsistency problems and data loss. The DTC Transaction List in the Component Services MMC allows you to monitor the transactions as they occur because it shows any distributed transactions that are currently executing. You can even resolve a transaction by right-clicking it and forcing it to commit or abort from the options on the pop-up menu. The DTC Transaction Statistics will let you view the activity statistics running on the system. If your system is having performance issues,

this feature can assist you in determining whether distributed transactions need to be given a larger-capacity system.

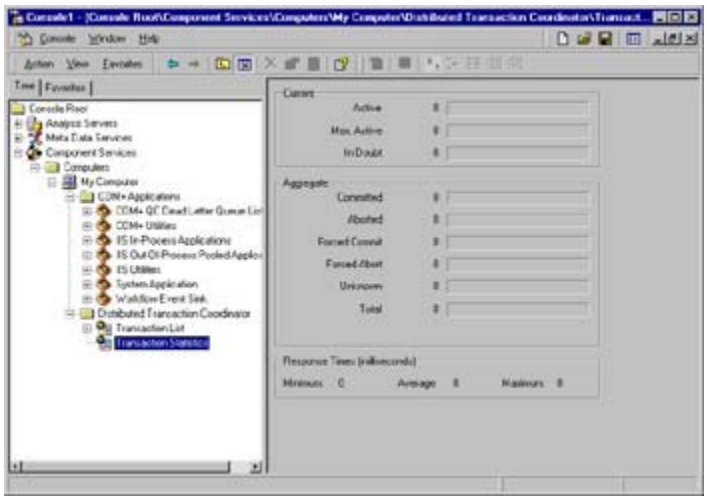


Figure 6.24: DTC Transaction Statistics in the Component Services MMC.

Metadata is the data that is used to describe another set of data. It is generally considered an indexing method using a summary and details about a database within the SQL Server system. For example, you might have a table that lists all Internet users' shipping information. The summary information used to describe that data, such as shipping costs by state, is considered metadata. Metadata is used most within the Analysis Services.

The Meta Data Services console is useful in viewing the metadata listed for a database. To use the Meta Data Services MMC, you must first start by registering a database. To do so, right-click the root of the console, and select Register Database. After the registration is complete, you can use the Meta Data Services console to view metadata.

The Analysis Services MMC, shown in Figure 6.25 , is used for discovering the statistics about analysis of the database. When you install an analysis server, this console is a standard option. You can access it by clicking Start | Programs | Microsoft SQL Server | Analysis Services | Analysis Manager.

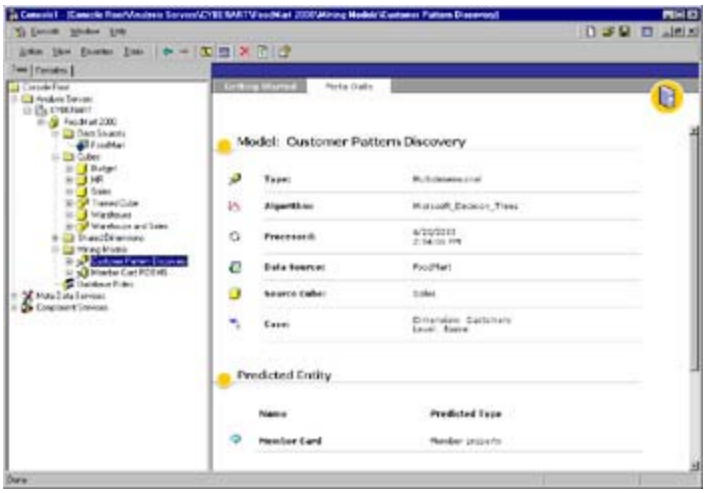


Figure 6.25: The Analysis Services MMC.

Moving and Copying SQL Server Databases

On occasion, administrators find that they must move or copy a database from one server to another. The processes for both are identical. The only difference between a move and a copy is that the original database is deleted in a move. The following method can be used for either copy or move operations:

1. Back up the database on the source computer.
2. Back up the database a second time, and verify the backup.
3. Install an instance of SQL Server on a destination computer.
4. Configure backup devices on the destination computer.
5. Review the files on the destination computer to ensure that no duplicate filenames exist. If there are existing files with the same names on the computer and you do not allow those files to be overwritten, there will be an error. If you do allow those files to be overwritten, the files will be overwritten, which might not be desirable.
6. Review the file and directory structure to ensure that drive letters and folders exist where you will be restoring files. If you restore a file that is located on a nonexistent drive location, there will be an error.
7. Ensure that the destination computer has Full-Text Search installed and the Microsoft search service started, if the database being copied uses full-text indexing.
8. Restore the backup to the destination computer. The restore proc-ess will automatically create the appropriate database files.
9. Validate that the database is accessible by clients.
10. If you are moving the database, locate the database in Enterprise Manager on the source computer, right-click it, then select Delete from the pop-up menu.

You can use Transact-SQL statements to restore database files. The only person logged in to the database should be the login account executing the restore process—typically the SA account. The following shows the T-SQL statements for restoring database files:

```
USE

master


GO



RESTORE DATABASE databasename
   FILE='databasename_data_1'
   FILEGROUP='filegroupname'
   FROM backupdevice
   WITH NORECOVERY,
   REPLACE
GO
```

```
RESTORE LOG

databasename


        FROM

backupdevice


        WITH NORECOVERY
GO
RESTORE LOG

databasename


        FROM

backupdevice


        WITH RECOVERY
GO
```

Note When you restore a database onto a different computer, the owner of the database is not the owner of the original database on the source computer. Instead, ownership is assigned to the SQL Server login account that initiated the restore operation. The system administrator (SA), as well as the new owner, can change the ownership if another owner is desired.

The REPLACE statement specifies that files can overwrite existing files in the same location with the same name. The NORECOVERY statement states that files have been modified since the last backup. RECOVERY is used if files have not been modified since the backup. For those files that have been modified, use the RESTORE LOG statement to apply the transaction log backup. When using RESTORE LOG, use NORECOVERY if there are further logs to apply, but use RECOVERY if there are no further logs to apply.

If you are moving the files to new locations, the following example moves a specific set of files and uses the MOVE statement to move files to new locations:

```
USE

master


GO



RESTORE FILELISTONLY
        FROM
```

```
backupdevice


RESTORE DATABASE

databasename


        FROM

backupdevice


        WITH NORECOVERY,
        MOVE 'database_datafile_1' TO
'C:\Folder\database_datafile_1.mdf',
        MOVE 'database_datafile_2' TO
'C:\Folder\database_datafile_2.mdf'
GO
RESTORE LOG

databasename


        FROM

backupdevice


        WITH NORECOVERY
GO
RESTORE LOG

databasename


        FROM

backupdevice


        WITH RECOVERY
GO
```

Enterprise Manager

You can restore files with Enterprise Manager. To do so, follow this procedure:

1. Log on as a user with administrative privileges.
2. Click Start | Programs | Microsoft SQL Server | Enterprise Manager.
3. Navigate to the server, and expand it.
4. Expand the Databases container below the server.
5. Right-click the database.
6. Select All Tasks from the pop-up menu.

7. Select Restore database.
8. Type in the name of the database that you will restore.
9. Select the option “From device,” as shown in Figure 6.26 .

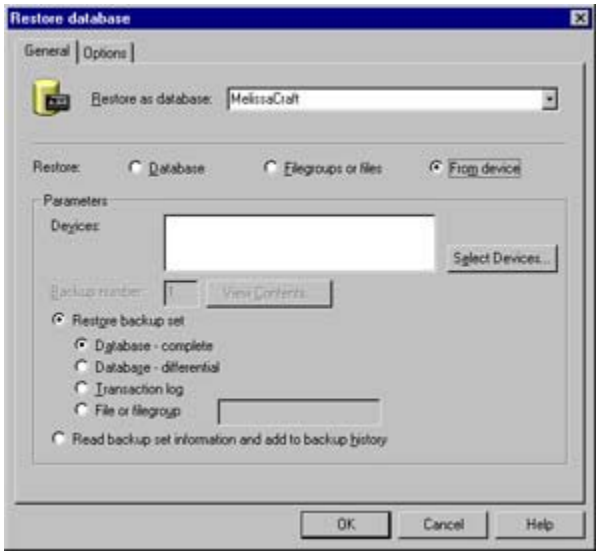


Figure 6.26: The Restore database dialog box.

10. Click Select Devices.
11. Select Tape or Disk and then the device where the backup exists.
12. In the Restore Database dialog box, click View contents, and select the backup set that you will restore—or to save time, simply type in the name of the backup set.
13. On the Restore backup set dialog box, click File or Filegroup, and type the names of the files that will be restored.

Copy Database Wizard

The most convenient way to copy or move a database is to use the Copy Database Wizard. Permissions are important in this operation; you must be an SA on both the source and destination SQL Servers. This method does not require server downtime, and to avoid performance issues, the operation can be executed on a scheduled basis to avoid times when the database undergoes a heavy workload. Another advantage is the easy-to-use GUI Wizard.

The Copy Database Wizard is easy to locate. In Enterprise Manager, click the Tools menu, and select Wizards. (Alternatively, you can right-click a server, select All Tasks, and then select Copy Database Wizard from the pop-up menu.) The screen that you will be presented with will have four expandable nodes. Expand the Management node, and select the Copy Database Wizard, as shown in Figure 6.27 . Then click OK.

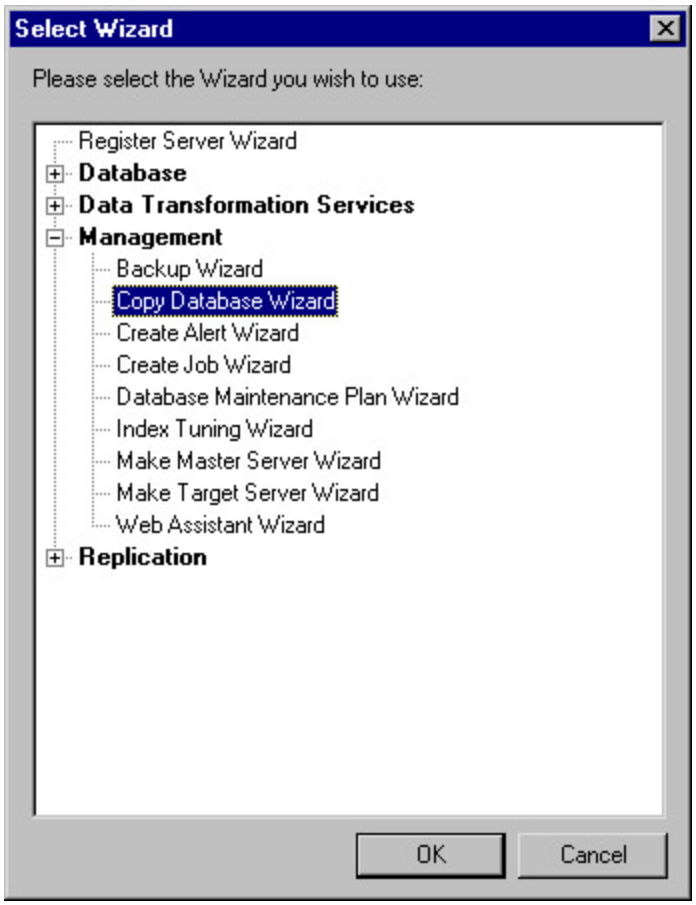


Figure 6.27: The Management Wizards list.

You should have administrative privileges to run the Copy Database Wizard. When you first start the wizard, you will be presented with a Welcome screen. Click Next to continue. The next dialog box with which you are presented allows you to select the SQL Server that contains your source database. You can accept the default, type in a name, or click the ellipsis (…) button to search for another source server. Then select the authentication method that you will use (either Windows NT authentication or SQL Server authentication), and click Next to continue.

The next screen prompts you to select a destination server. Notice that the default selection is <local>, illustrated in Figure 6.28 . This is the reason for executing the Copy Database Wizard on the destination computer. Again, select whether to use Windows authentication or SQL Server Authentication, and click Next to continue.

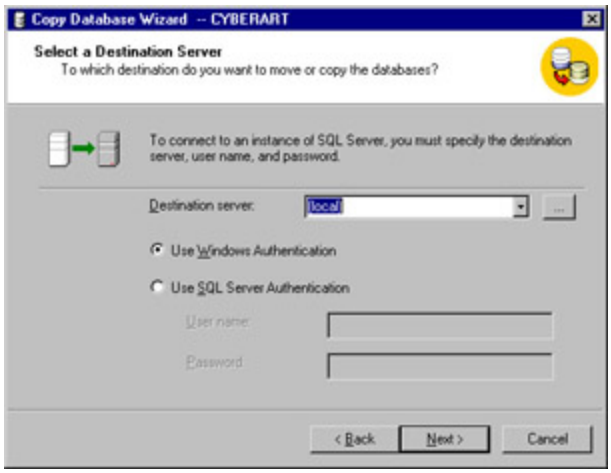


Figure 6.28: The default destination server is <local>.

Warning Source and destination servers cannot be the same when you use the Copy Database Wizard. You will not be able to copy a system database, such as msdb or master, with the Copy Database Wizard, either. If you want to execute a special copy or move operation, you can use Data Transformation Services instead.

Your next task is to select the databases that you want to move or copy. You will not be able to move or copy databases that have the same name, nor will you be able to rename the database during the move or copy operation. For each operation, you can mark either the move or copy column to the left of the database in the dialog window, as shown in Figure 6.29 , and then click Next to continue.

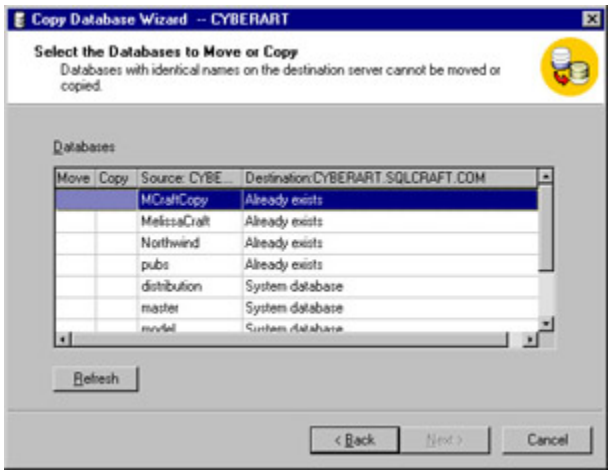


Figure 6.29: Select the Databases to Move or Copy.

Select any other objects to move or copy on the next screen. Then determine the schedule for the operation.

Once you have completed the Copy Database Wizard, the information that you selected is named and saved as a package on the destination server. This package is saved regardless

of whether it is executed immediately or awaiting a scheduled time and date for execution.

Tip To ensure consistency, make certain that there are no active sessions on the server before you begin the move or copy operation. An active session will prevent the Copy Database Wizard from running.

Detaching and Attaching Databases

If you have multiple instances of SQL Server, you can detach the data and transaction log files of a particular database from one instance and then attach them to another server or even another instance of SQL Server on the same computer. The result of this operation is that the database is attached to the new server, completely intact and in the same condition as when it was detached from its former server. This is an excellent method for moving a database without having to restore a database backup or move the database to a newer, larger storage system.

If you move a database from one physical SQL Server system to another, you first detach the database from the source server. Then you move the database files to the new server. Finally, you attach the database to the new server, indicating the new location of the database files. Additionally, you should remove replication for a detached database. This is done with the sp_removedbreplication statement. You can attach and detach databases using Enterprise Manager as follows:

1. Log on as an account with administrative privileges.
2. Click Start | Programs | Microsoft SQL Server | Enterprise Manager.
3. Navigate the server group to the server that holds the database in question.
4. Expand the server.
5. Expand the Databases container.
6. Right-click the selected database.
7. Select All Tasks from the pop-up menu.
8. Select Detach Database. Click OK in the confirmation dialog box, shown in Figure 6.30 .

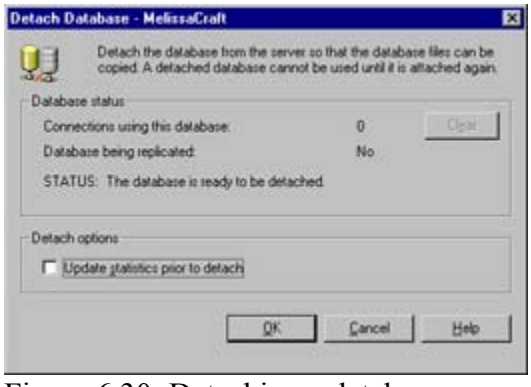


Figure 6.30: Detaching a database.

9. Click OK to the message that the detaching was completed successfully. When you return to Enterprise Manager, you will see that the database no longer shows up under the Databases container.
10. To attach a database, right-click the Databases container in Enterprise Manager where the new database will be attached.
11. Select All Tasks.
12. Click Attach Database. You will see the dialog box shown in Figure 6.31 .

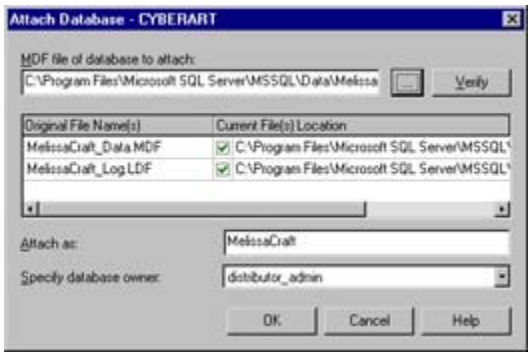


Figure 6.31: Attaching a database.

13. Click the ellipsis (…) button to search for the database file, and when it is found, click OK. Or if you know the name of the file, type it into the space provided.
14. Click OK.
15. Click OK to acknowledge the message that the attachment was successful. The database will now appear within the Databases container.

When a database is detached and later attached to another SQL Server in the same Active Directory domain or another Active Directory domain within the same forest, the permissions within the database do not change. Outside the database, such as the right to logon to the server itself, however, permissions need to be reapplied. Therefore, you need to apply the right to logon to the server to all database users, or they will be denied the right to access the server, even though they have the appropriate permissions *within* the database itself.

Linked Servers

Object Linking and Embedding (OLE) is the method used for accessing outside databases, especially in running distributed transactions. One task that you must perform, however, is creating the linkage, or *linked servers*. Once linked servers are created, you can conduct distributed transactions, referencing rowsets from an OLE DB data source for queries and data operations.

Linked servers consist of an OLE DB provider and an OLE DB data source. The OLE DB provider is a *dynamic link library (DLL),* which interfaces with the data source. The OLE DB data source specifies which specific database will be accessed. OLE DB providers exist for databases, text files, spreadsheets, and query results.

When a client application queries data with a distributed transaction through a linked server, the SQL Server transmits the rowset requests to the OLE DB and executes the remainder of the command locally, following the DTC coordination rules. Rowset requests can be a query or request for an entire table of data.

Linked servers can be defined within SQL Server Enterprise Manager or with stored procedures. To use Enterprise Manager, follow these instructions:

1. Log on with administrative privileges.
2. Click Start | Programs | Microsoft SQL Server | Enterprise Manager.
3. Expand the SQL Server group.
4. Expand the SQL Server.
5. Expand the Security container.
6. Right-click Linked Servers.
7. Select New Linked Server from the pop-up menu.
8. You will see the dialog box shown in Figure 6.32 . Type in the name for the linked server, click Provider Options, select the options for the OLE DB link, and click OK.

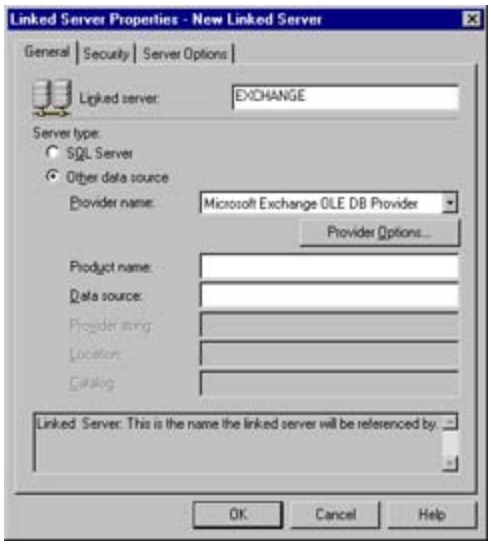


Figure 6.32: Linked Server dialog.

9. Complete the remaining information about the linked server, and then click the Security tab.
10. Type in the login to be used to access this OLE DB and any other login information required by the linked server.
11. Click the Server Options tab and select the options for this OLE DB.
12. Click OK to create the linked server.

You can also use stored procedures, such as the statement *sp_addlinkedserver,* to define a linked server. You can use *sp_linkedservers* to view infor-mation about the OLE DB data sources. To remove a linked server, you use *sp_dropserver.*

Distributed Queries

Once you've gone to all the trouble to create linked servers, you can start using those links to your advantage by executing distributed queries. *Distributed queries* are the searches that access the data from multiple data sources, regardless of the location of those data sources.

As discussed in the previous section , OLE DB data sources provide data in a form called a *rowset* . The rowset is a tabular object that can be referenced by T-SQL statements, just as though the rowset were a table located on a SQL Server. This capability is especially useful if you have several departments running different types of databases and then you need to create an enterprise report based on data adjoined from different databases.

Database Maintenance Tools

Database maintenance is a mission-critical job in today's information- gathering society. For instance, on the Internet, databases provide the means of obtaining and retaining shipping information for online shoppers. When an online shopper returns to the same site, if the shipping information is retained, the shopper has less data to enter in order to purchase an item. So, it appears that both the shipper, who retains accurate location information, and the shopper, who needs less time to conduct online shopping, benefit from this data. However, if the shipping information database were to call up the wrong data, fail altogether, or take several minutes to return the data to the online shopper (in other words, indicate performance issues), the shopper will be likely to cancel the order and the shipper to lose business. The bottom line is, database maintenance is mission-critical.

DBCC

Database Console Commands (DBCC) are statements developed to check the physical and logical consistency of the SQL Server database and perform administrative tasks such as file management. DBCC statements are capable of fixing problems that are detected. Some of the more popular DBCC maintenance statements are listed in Table 6.2 .

Table 6.2: DBCC Statements

Statement	**Function**
DBCC CHECKALLOC	Checks the allocation of pages in specified database.
DBCC CHECKCATALOG	Checks for consistency of system tables within a database.
DBCC CHECKCONSTRAINTS	Checks the integrity of constraints on a table.
DBCC CHECKDB	Checks the integrity of everything within the database.
DBCC CHECKIDENT	Checks and corrects the identity of the specified table.
DBCC CHECKFILEGROUP	Checks the integrity of a specified filegroup.
DBCC CHECKTABLE	Checks the integrity of data, index, text, ntext, and image pages.
DBCC CLEANTABLE	Reclaims space used by varcher data.
DBCC DLLNAME	(FREE) Unloads the DLL from memory.
DBCC DBREPAIR	Repairs the database.
DBCC DROPCLEANBUFFERS	Removes clean buffers from the buffer pool.
DBCC FREEPROCCACHE	Removes everything from procedure cache.
DBCC INDEXDEFRAG	Defragments the index.

Table 6.2: DBCC Statements

Statement	Function
DBCC INPUTBUFFER	Returns the last statement sent by a user associated with the identified system process id (SPID).
DBCC OPENTRAN	Returns information about the oldest active transaction on the database.
DBCC OUTPUTBUFFER	Returns the output sent to SQL Server by a user associated with the specified SPID.
DBCC PINTABLE	Marks a table to be pinned or held in the cache.
DBCC SHOWCONTIG	Displays information about fragmentation.
DBCC SHOW_STATISTICS	Returns statistics for the target on the specified table.
DBCC SHRINKDATABASE	Recaptures lost space within all the files associated with a database.
DBCC SHRINKFILE	Recaptures lost space within a specific file, either a database file or a log file.
DBCC SQLPERF	Returns information about the transaction log space use.
DBCC TRACEOFF or TRACEON	Traceoff disables the trace flags; traceon enables the trace flags.
DBCC TRACESTATUS	Returns the status of the trace flags.
DBCC UNPINTABLE	Marks a table to be unpinned or treated normally in cache.
DBCC USEROPTIONS	Returns the status of current connection SET statements.
DBCC UPDATEUSAGE	Detects and corrects inaccuracies of the sysindexes table.

Database Maintenance Plans

Database maintenance plans are a method of ensuring that a database is optimized and performing well. SQL Server 2000 includes a Maintenance Plan Wizard for use in creating a database maintenance plan.

Maintenance Plan Wizard

The Database Maintenance Plan Wizard creates a job, which includes multiple maintenance tasks. The job is then scheduled to perform those maintenance tasks periodically. You can initialize the Database Maintenance Plan Wizard from within Enterprise Manager:

1. Log on as an account with administrative privileges.

2. Click Start | Programs | Microsoft SQL Server | Enterprise Manager.
3. Navigate to the SQL Server Group.
4. Click the Tools menu.
5. Select Database Maintenance Planner.
6. The wizard will start. Click Next to bypass the Welcome screen.
7. Your first task is to select the databases that will be maintained, as shown in Figure 6.33 . After selecting the databases, click Next.

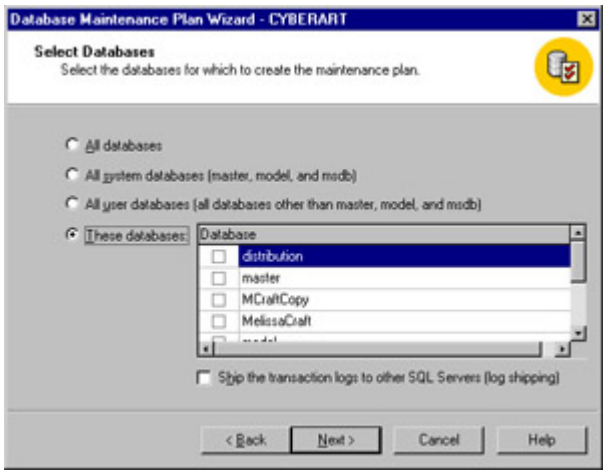


Figure 6.33: Select databases to be maintained.

8. The next screen allows you to optimize the way that data is organized in the databases. Select the options you want to use, accept the default schedule or change it, and click Next.
9. You are prompted to state your preference for a database integrity check, as shown in Figure 6.34 . Accept or change the schedule, and click Next.

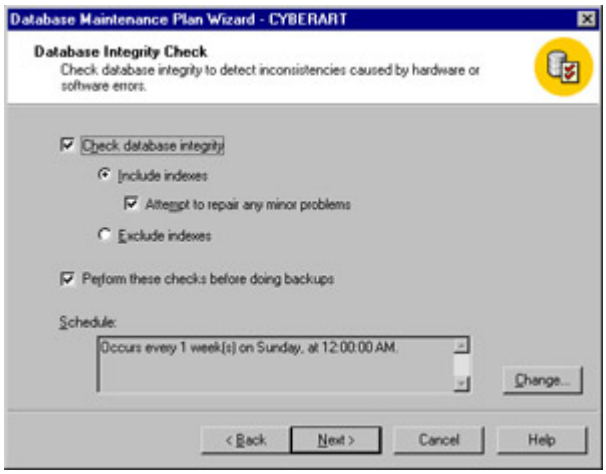


Figure 6.34: The Database Integrity Check dialog box.

10. The next dialog box specifies the database backup plan. After configuring these options and the schedule, click Next to continue.

11. If you selected a backup to a disk, you will be prompted to state the directory where the backup files will be placed. Click Next after configuring.
12. Then you are prompted to configure the Transaction log. Configure this option, and click Next.
13. You can then configure reports regarding the maintenance tasks. Configure this option, and click Next.
14. Maintenance Plan History is stored in a table on the SQL Server. You are given the option to specify the server where this table is located. Make your selection, and click Next.
15. Finally, you are provided with a summary screen, displayed in Figure 6.35 . Verify that it shows the options that you selected, type in a new name for the maintenance plan if desired, and click Finish.

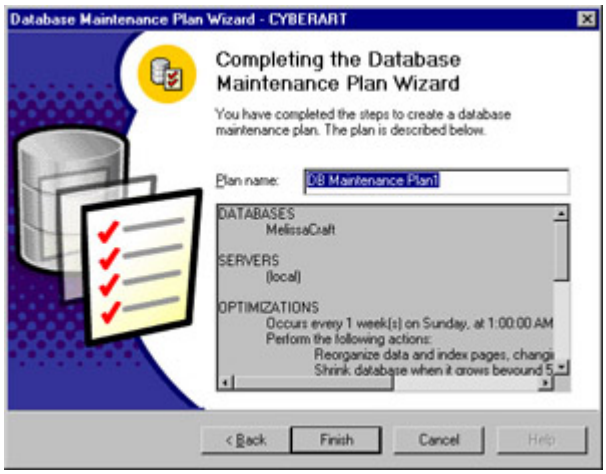


Figure 6.35: Summary of the maintenance plan.

SQL-DMO

SQL Distributed Management Objects (SQL-DMO) is an API that you can use when creating stored procedures on a SQL Server and custom administration tools. In general, DMO alters the way that a SQL Server component is viewed, so you can use the component's attributes as though they were an object instance. This feature lets you use object methods in automating administration.

SQL-DMO is actually a single DLL—sqldmo.dll. Other files, such as sqldmo.h, which is a C++ header file, are provided to assist in development tasks. SQL-DMO is built on the Component Object Model (COM), which means that any COM-compatible programming language can be used with it to create administrative tools.

Many operating systems support various methods for automating administration. It is common to find unique scripting language support meant just for this purpose. In SQL Server, SQL-DMO was developed to allow a DBA to use any COM-based language, such as Visual Basic or C++, that the DBA was comfortable using.

SQL Enterprise Manager is built on SQL-DMO, which means that anything you do within Enterprise Manager can also be done using SQL-DMO. This is a powerful statement representative of the kinds of objects and attributes that SQL-DMO defines.

For example, the SQL Server is described by a SQLServer object. That is further related to objects that represent databases and tables. Each object branches off from the root SQLServer object in a hierarchical fashion. This indicates that you cannot reference an object unless you've instantiated the entire hierarchy of objects, all the way from the SQLServer object leading to the object. The hierarchy is further organized with *collections* , which are groups of same-type objects that share the same parent object. Collections reduce the time required to create administrative tools, since you do not need to specify the name of each object but can simply reference the collection instead. For example, the Tables collection contains a set of tables.

Managing a SQL Server

When you are called on to manage a SQL Server, you sometimes don't know where to start. Here are some hints:

1. Design the SQL Server placement of database files so that tempdb is on a different disk and so that your disk reads and writes for tempdb are separated from those for the other databases.
2. Steer clear of batch jobs or queries that hit the CPU hard. If you must run them, run them during nonproduction hours.
3. Plan to run CHECKDB during off hours only. The DBCC statement CHECKDB uses the most processing power, since it examines everything within the database. Not only is this a processor-intensive command, but it also takes a long time to run—especially if the database is very large.
4. When you run DBCC commands, especially the CHECKDB command, make sure that you aren't running a backup of the database at the same time. In fact, avoid running DBCC commands any time anyone is running a transaction, either. Avoiding this situation will increase the performance of the DBCC.

Automating Administrative Tasks

As server farms grow to incorporate more database servers and storage space, DBAs find themselves relying increasingly on automating administrative tasks. With SQL Server 2000, DBAs can create jobs and set alerts to assist in the management of the server.

SQL Server Agent

The SQL Server Agent controls multiple areas of database administration. It not only manages replication, as we discussed previously, but it maintains error logs, runs jobs, and triggers processes based on alert settings. SQL Server Agent components include:

- **Alerts** These are actions that are triggered when a specified event occurs. Usually, alerts are set to monitor for errors and take action if a specific error occurs.
- **Jobs** A job is an object that contains a set of actions to be taken. Jobs can be scheduled to occur at times when an administrator is not physically present to execute them. SQL-DMO can be used to create jobs.
- **Operators** An operator is an account that has been identified to receive e-mail or pages. Operators are usually notified only when an alert kicks off.

Alerts and Operators

You can manage both alerts and operators from within SQL Enterprise Manager.

SQL Mail

One of the critical components to ensuring that an operator receives an alert is e-mail. SQL Mail provides the means to create and send those critical e-mail messages. SQL Mail does not, itself, deliver e-mail to a recipient. Instead, it provides a means to generate an e-mail message, even one that contains the results of a predetermined query, and establish a client connection to an e-mail server in order to transmit the message. The client connection that SQL Mail establishes is an extended *Messaging Application Programming Interface (MAPI)* connection. MAPI is a standard API used with many e-mail systems, including Microsoft Exchange.

When you configure SQL Mail, you need to have a connection to the e-mail server, a mailbox, and a user account that is capable of logging on to the SQL Server. To automate SQL Mail, you must create a job that uses a stored procedure called sp_processmail. This procedure checks the mailbox for any mail and kicks off xp_sendmail to execute queries in the e-mail text and forward the result to the recipients.

Several stored procedures take advantage of SQL Mail. When a SQL Mail-activated stored procedure executes, SQL Mail requests the defined mail profile for the domain account that triggered the stored procedure. The account for which you will configure a mail profile is the startup account that is listed in the SQL Server properties on the

Security tab. This service account must have a mailbox configured on an Internet mail server or Exchange Server. You can create a mail profile this way:

1. Log on to the server as the service account.
2. Open the Mail icon in the Control Panel.
3. Select the appropriate mail server, whether Exchange Server (if available) or an Internet mail server.
4. Complete the appropriate information to configure the profile for the account.

To implement SQL Mail:

1. Log on as an account with administrative privileges.
2. Click Start | Programs | Microsoft SQL Server | Enterprise Manager.
3. Navigate to your selected SQL Server and expand it.
4. Expand Support Services.
5. Right-click SQL Mail.
6. Select Properties from the pop-up menu. You will see the dialog box shown in Figure 6.36 .

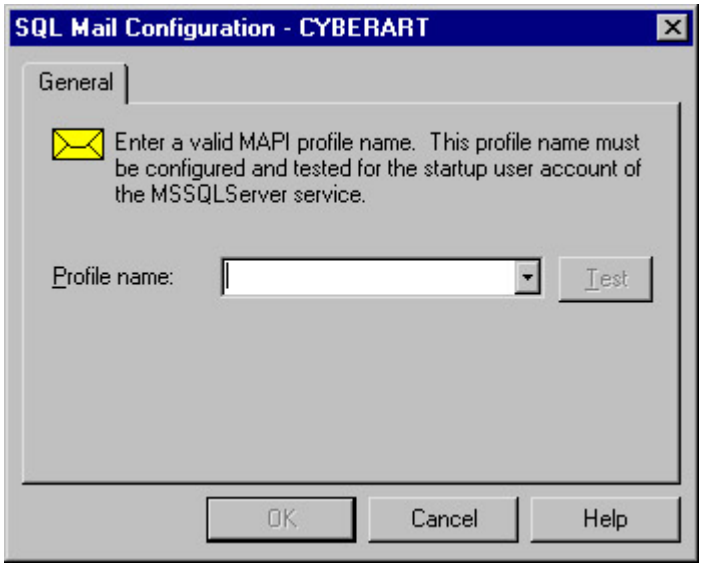


Figure 6.36: The SQL Mail Configuration dialog box.

7. Type or select a name from the drop-down list, then click Test.
8. Click OK to close the dialog box.

Setting Up Operators

You can create new operators in SQL Server Enterprise Manager. Start by expanding SQL Server and then the Management container below it. Next, expand the SQL Server Agent container. Right-click the Operators icon, and select New Operator from the pop-up menu. You will see the dialog box presented in Figure 6.37 . Complete the user information on the General tab, and state the pager schedule. Click the Notifications tab.

Below the “Notifications sent to this operator by” option, check one or more of the check boxes to send an alert. You have the option of sending alerts via e-mail, via pager, or through a net send message. You can check any or all of these options for each alert listed.



Figure 6.37: Operator properties.

Defining Alerts

Several default alerts are automatically defined within SQL Server. However, most of them are demonstration alerts, and the remainder apply to replication. You must define an alert before you can send a notification. To define your own alerts, you use SQL Server Enterprise Manager:

1. In Enterprise Manager, navigate to the selected SQL Server.
2. Expand the Management container.
3. Expand the SQL Server Agent container.
4. Right-click Alerts.
5. Select New Alert from the pop-up menu.
6. Type in a name for the alert. Note that the name must be unique.
7. Select the event to which the alert will respond. You can also specify an error number.
8. Select the Database that will trigger this alert.
9. If you want to restrict the alert to a particular text string, you can type in the text string in the box next to “Error message contains this text.” This is helpful if you are troubleshooting a particular error.
10. Click the Response tab, and select a job to be executed, if you want to trigger a job when the error occurs.
11. Select the operators to notify and the method of notification (e-mail, pager, net send).
12. Type in any additional message to send in the bottom text area, then click OK to complete the process.

Summary

Active Directory provides a directory service that stores information about users, computers, groups, applications and other network services, accounts, and resources. SQL Server 2000 integrates with Active Directory, not only using it for Windows NT authentication but also being able to register its own information within Active Directory for use by end users and applications that are Active Directory-aware.

Before listing any other objects in Active Directory, the administrator must begin by registering the SQL Server within it. This creates in Active Directory a new object describing the SQL Server. Next, the administrator can register SQL Server databases, SQL Server publications, and SQL Server Analysis Services in Active Directory. A user or an application can then conduct a search for a SQL Server component using information about the component. The user or the application is not required to know the SQL Server instance on which the component resides. This means that an administrator can register SQL Server components, then move them physically around the network without having to change the information in multiple client applications.

There are many ways to administer various SQL Server elements. Because SQL Server can utilize Windows NT authentication, a SQL Server administrator might be required to create users within Active Directory. This is done using the Active Directory Users and Computers console. In addition, the administrator might want to manage the information listed in Active Directory. This can be done using the ADSI Edit or LDP tools, both of which are available in the Support\Tools directory on the Windows 2000 Server CD-ROM.

Much of the administration for SQL Server is handled in the Enterprise Manager application; however, Analysis Services are administered in the Analysis Manager application. Both of these consoles and all the Active Directory consoles are Microsoft Management Console (MMC) utilities. The MMC is a common framework from which to run snap-in utilities; it provides a consistent interface for an administrator.

Moving and copying databases can be accomplished in many ways. Traditionally, DBAs use a backup and restore method between servers. However, within Enterprise Manager is also a Copy Database Wizard that can simplify this process as well as execute while the database is online. Moves are easily managed through the process of detaching a database from one server, conducting a file move procedure, and then attaching the database to the destination server.

One of the strengths of SQL Server is the ability to use Object Linking and Embedding (OLE) to query data existing in non-SQL Server data repositories. When OLE is used, a linked server is created. Then distributed queries are executed against the linked server. The data within the linked server is considered a rowset and provided in tabular format.

Maintenance and administration of the SQL Server can be conducted using automated methods. An administrator can execute the Database Maintenance Plan Wizard to select

the types of maintenance to be executed and the schedule on which to execute them. The administrator can also configure SQL Server Agent alerts, jobs, and operators. Once configured, a SQL Server can encounter an error that triggers an alert. The alert will trigger a job and send a notification through SQL Mail to an operator.

FAQs

Q. We want to configure SQL Server to list publications in the Active Directory for internal users. However, we do not want our external SQL Server users to be listed as accounts in Active Directory. Do we also need to use Windows NT authentication for user logins, or can we use SQL authentication?

A. You may use SQL authentication for logins plus be able to list your publications in Active Directory. The two are not mutually exclusive.

Q. When copying a database from one server to another, I detached the database. Now I cannot find the files to move to the other server. What should I do?

A. Most database files are located in the default Data folder and are named with the same name as the database. The default location for these files is C:\Program Files\Microsoft SQL Server\MSSQL\Data. Select the file with the .MDF extension and the same name as the database you are moving.

Chapter 7: SQL Server Backup and Recovery

Introduction

Backup and recovery procedures are essential to every organization, especially those employing central database systems. Recovering from system failure or user error can mean the difference between a business inconvenience and a profit loss! SQL Server 2000 offers several backup methods and media storage options that will allow you to implement a backup strategy that works within your organization.

Via support for snapshot backup and recovery options, performing online backups while minimizing the impact on server performance is essential for the 24 x 7 availability required in many e-commerce and multinational organizations. For large database backups, using multiple backup devices can speed the backup process and increase system availability. Combined with backup verification and scheduling capabilities in SQL Server, your data can be secure and available for recovery in case your database server is not.

This chapter reviews the SQL Server backup models as well as the steps to planning and implementing a backup and recovery strategy. You will perform a backup and restore procedure using the Backup Wizard and T-SQL statements. The end of this chapter reviews developing an ongoing test strategy to ensure that your backup procedures are accurate and can be implemented in a disaster recovery situation.

Planning and Implementing a Successful Backup and Recovery Strategy

The importance of a thorough and practiced database backup and recovery strategy will become evident the first time you need to implement your recovery procedures—in other words, the first time your server goes down and you lose your database. A successful recovery strategy begins with planning your backup strategy, even before your database has been implemented.

Determining Data Recovery Requirements

The way your backup strategy is designed is based on the requirements for your data recovery. These requirements can simply be formulated as "How much information are you willing to lose, and what is the maximum time available for recovery?" These questions boil down to financial costs. The perfect solution is one of no loss of information and recovery in zero seconds. Such a solution requires significant investments in hardware to accomplish this level of data protection. The result is choosing the optimal solution for your organization, meaning that your recovery strategy balances the costs of losing information and the investments you need to make to recover as quickly and with a little data loss as possible. In fact, it all has to do with *risk management* and *contingency planning* .

Backup and Recovery Strategies

A well-established backup and recovery strategy is essential to your organization. This is also the case for the backup of your Windows 2000 installation and registry, applications, Active Directory database, and user data. However, you should head for the data vault to retrieve the tape only if all other options fail.

The architecture of the complete IT infrastructure should be designed and implemented with a high level of redundancy offering fail-over capabilities, especially if you are running 24 x 7 applications like many of today's Web applications. Within this kind of infrastructure, you should consider a number of options. The first is redundancy of the database server using solutions such as Microsoft Cluster Services, so your database servers fail over in the event of system problems and your applications can continue to operate. Within your server, all parts such as network cards and power supplies should be in a redundant configuration, which guarantees continuity of operation.

A second option in maintaining SQL Server availability is the use of SQL Server 2000 capabilities such as log shipping and replication. These techniques safeguard your system from unnecessary loss of data or availability by having warm or replicated standby servers.

The third option involves dispersing your data across multiple drives. Never configure backup files stored to disk, transaction logs, system databases, and user-defined database files on the same physical disk. Besides offering performance improvements, a dispersed configuration increases the restore capabilities and recovery time over a single physical disk configuration.

Remember that relying on media backups is the same as accepting loss of data as media backups will never contain the most recent data in online transaction processing systems. Since the loss of data is unacceptable in most cases, maintaining your servers for high availability should be your goal. Having to go to your backups to recover your server should be your last option.



There are three types of problems in which the need for recovery is necessary: permanent loss of one or more servers due to a disaster (natural or manmade); failure of storage devices—for example, a disk crash or power failure; or invalid data modifications (human error)—for example, a content update batch run twice, making a large number of records invalid.

Frequency of Database Changes

Your organization will rely on the database backups you made if one or more databases become unavailable or beyond repair. To what extent you can recover from this failure has to do with how fast the data changes in the database. A customer relationship management (CRM) database or database warehouse will not change that much over the course of a day, whereas an electronic commerce system or other online transaction processing (OLTP) system can be subject to numerous changes every second. If you are able to recover only up to an hour before failure in both cases, the latter situation will carry grave consequences, while the first will probably have no consequences at all.

The second issue is the possibility of manual reconstruction of lost data. Do the logs or other resources available make it possible to reconstruct the changes that where made between the moment of database failure and the point to which the database could be recovered? In the case of the CRM database, you probably have e-mails or notes that enable you to reenter the changes. For the electronic commerce system, however, that is most likely not the case. If you look at the effect of the frequency of database changes on the three types of problems, you will recognize that a strong backup strategy is crucial.

Cost of Data Loss and Availability

The frequency of changes to a database is a leading factor in your backup strategy in terms of cost and availability. Due to unrecoverable data loss or nonautomatic recovery of data, a number of costs can be determined:

- **Manual reconstruction of the lost database changes** This cost can be expressed in the number of hours one person is busy reentering lost information. This operation is very work intensive can take hours or days to complete.
- **Lost business orders or transactions** If you are running an e-business, losing data means permanently losing orders or transactions. It is very likely that you will never know who placed an order if you do not take measures to be able to reconstruct orders.
- **Goodwill** Since bad news always sticks, customers can become wary of doing business with you if they run into problems with your servers. This can have long-term effects and not only costs the loss of business but also the cost of public relations to reestablish your company's reliability.

Even if you are able to recover with nearly zero data loss, it can take some time before the recovered database becomes available again. During this time, the applications that rely on the database cannot be used, resulting in a number of additional costs:

- **Lost office hours** The employees that normally rely on these applications will be simply hanging around waiting for you to bring the data-base back online. For mission-critical applications, this lost work time, even if only a few hours, can easily run into four-digit figures.
- **Lost transactions or orders** While the applications are offline, it is not possible to enter new transactions or orders. Based on the average volume and value of orders per hour, you can calculate the loss of profit.
- **Goodwill** This is the same cost as with data loss.

The message is clear: The costs of database failure and the subsequent recovery can run into very large numbers—or even run a company into the ground, if that company lacks a perfect backup and recovery strategy for its mission-critical applications. Devising a strategy for a single database can be straightforward; things can get more complicated if a database has relationships with other databases—for example, in a publish-and-subscribe configuration. If you cannot make a full recovery to the point of failure, the related databases must be synchronized (rolled back or even rebuilt). This action will increase the recovery time and costs.

A way to perceive the issue of cost is to invest in reducing the costs that come with a loss of data and availability. After you have determined what the projected costs can be when faced with a loss of data, you can calculate what kind of technical improvements you can make to reduce these costs. If the investments for these improvements are less than the costs, it is a worthwhile improvement. Nevertheless, you should always be prepared for the challenges that await you in terms of recovering data.

Note There's a well-known saying: "Bad news travels fast." Research has come up with some interesting numbers to prove this saying to be very true. On average, people tell their good experiences to 3 other people, whereas they tell 11 people their bad experiences. It does not take a scientist to figure out that one outage during the busiest hours of the day can undo all the hard work that your organization has done

to gain customer confidence and build your business.

Planning for Hardware Failure

Based on a standard setup of a database server configuration (see Figure 7.1), it is very likely that a hardware failure will trigger a database recovery. On the other hand, a hardware failure can also disrupt the database backup. In both cases, the failed hardware must be replaced swiftly. If replacement is not planned ahead, valuable time is lost. Planning also means that the server configuration on which the database runs makes it possible to replace separate hardware components quickly. The failure of hardware parts that affect the database backup and recovery are the tape unit, the disk unit, the server, and the network.

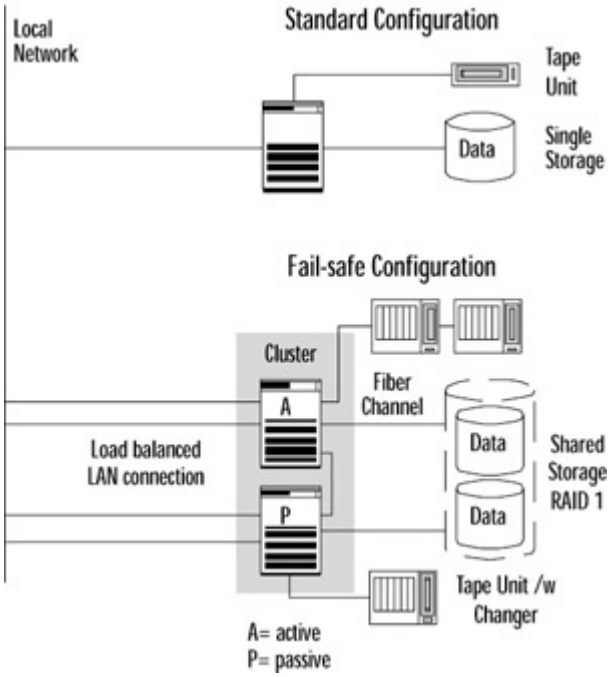


Figure 7.1: Standard versus fail-safe server configurations.

The Tape Unit

During a regular backup of the database, the tape unit can fail. The first action that must be taken is to reissue the backup action to another storage unit (tape or disk). Then the tape unit must be replaced with one that has the same media characteristics, so the backup procedure, including the media sets in use, can run without modifications. It might seem obvious, but make sure the replacement uses the same media (format), preventing media sets from becoming unreadable and making a restore impossible.

If a tape unit failure is determined, *always* check the backups of the previous days. It is likely that the tapes will return read errors, meaning that the tapes must be replaced or at least reformatted, and an emergency full backup of the database must be made. Clean the

heads of the tape unit regularly; doing so can prevent read and writer errors. Nowadays most units signal the need for a head clean; never ignore this signal.

Another option is to have a second identical tape unit installed. During normal operations, you can balance the backup jobs over both units; during a failure, you can make all backups on the remaining unit. The tape unit can also fail due to problems with the adapter. When both tape units are connected to the same adapter, nothing is gained from having two units. Install each backup device on a separate adapter. Since not every tape units have a backup power supply, it is worth the effort to buy one that has.

The Disk Unit

Depending on the way you configured the disk unit, failure of one or more disks, interface adapters, or power units could result in the database becoming inaccessible. In any of these situations, you will very likely need to perform a recovery of the database. Replacing the failed parts could result in differences in hardware versions; so new drivers have to be installed. Make sure that the new components work together. If the failure of the disk unit is not related to a crash of a hard disk, it is worth making an effort to recover as much information as possible from the disks, especially if the database you must recover is large or the number of changes to the database is large. Even if your disk storage unit has built-in redundancy of parts, you should have replacements at hand. In addition, analyze what factors could have contributed to the failure. The cause could be a known weakness of the failed part or an environmental circumstance such as an unstable power supply/grid, which triggered the failure.

The Server

When you are running the database server in a cluster mode, recovering from complete server failure is automatic. In a SQL Server fail-over cluster using Microsoft Cluster Services, all server activity will be transferred to the remaining cluster member so that the damaged server can be taken offline and repaired. If your database server runs standalone, you need to replace it with one with at least the same CPU and internal memory capacity. Remember, you need to restore the complete Windows 2000 setup, including the SQL Server 2000 installation and settings. Make sure that you have made all the necessary backups.

The Network

In the situation in which the database server runs in a distributed environment, the network provides the essential means of communication. Failure within the network can break the communication chain, resulting in possible loss of data. The resiliency of the network is very important in keeping the database operational. Most modern servers are available with redundant network cards and a high availability network will utilize multiple switches and routers to balance traffice in the event of equipment failure.

Waiting for new hardware can take up a large part of your recovery period. To be sure that the database server of your mission-critical applications becomes operational in the shortest possible time; adhere to the following guidelines:

- Make every effort to maintain your servers and elminate the need to recover your database server from backups.
- Use a hardware configuration that has a high level of redundancy. Most modern servers can be configured with hot-swappable power supplies, network cards, disk drives and memory. You can use the following list as a guideline to high availability:
    1. Install and configure Microsoft Clustering Services (MSCS).
    2. Every server in the cluster has the same tape unit (preferably with a media changer) installed.
    3. The disk storage is at least configured in RAID 1, and the storage unit must have redundant controllers. Using FiberChannel enables you to physically separate server and storage units with high speed communication.
    4. Network and storage adapters must be installed redundantly so that adapter switching can be done quickly.
    5. All hardware must have redundant power supplies.
    6. The power installation of the computer room must have an Uninterruptible Power Supply (UPS).
- Use only industry standard (non-proprietary) hardware, unless you make sure that the hardware producer is able to replace it. Never use hardware that is no longer produced or serviced.
- Replace parts that have known defects or bugs right away. Never wait for a problem to occur.
- Use service agreements with a short time-to-repair cycle or that have replacement hardware onsite. Even with redundant configurations, carry out repairs as soon as possible.
- Make sure that in case of a disaster (fire, earthquake, or tornado) you will be able to recover the database on an offsite configuration. This means that up-to-date media sets are also kept offsite. This kind of procedure must be part of your contingency planning.
- Your full recovery procedure must be practiced at least twice per year.

Selecting a Backup Strategy

When you select a backup strategy, it is important to find the optimal one, meaning that the backup should take as little time as possible, with a minimal loss of data, and that a consecutive restore can be performed in a straightforward manner. Simply put, if you are not able to reliably restore a database from the backup you made, do not even bother making a backup. Since this scenario seems unlikely, we must review a number of backup options to get better insight into the optimal choice for a backup strategy.

Backup Strategy Options

As mentioned, the costs and frequency of data changes play a big role in determining the need to make backups. Once it is decided that a database must be backed up, the questions become: With what frequency must this be done, and which technique should be used? Since SQL Server 2000 can perform online backups, the backup can run while people are using the database, as long as:

- No database files are deleted or created.
- No "shrinking" (truncating) operation is performed on a database file.
- No indices are deleted or created.
- No nonlogged operations are performed.

In a situation in which one of these actions is started while a backup is running, it will fail. In the reverse situation, the backup will be aborted during the initialization phase.

Within the SQL Server 2000 backup and restore architecture, a number of backup/restore models exist. Depending on the model you choose when planning for a backup strategy, you can choose from among certain database backup options. Remember that the way a restore has to be executed is directly related to backup strategy you choose.

Note The terms *strategy* and *model* can be used interchangeably since you derive a backup strategy based on a backup-and-restore model.

Simple Backup and Restore

A *simple backup-and-restore* model refers to the periodical backup of your database. For example, SQL Server is configured to make a complete backup of the database every night. By restoring this backup, you roll back the state of your database to that of the day before. This model is applicable to databases that have a low number of changes during a given period *and* for which these changes can be reconstructed from other sources. If the time it takes to back up the database extends beyond the time to plan for it, you can also use the differential backup option, discussed later in this section.

Full Backup and Restore

The *full backup-and-restore* model should be implemented in environments in which one of the following is the case:

- The environment highly dependant on the information stored in the database.
- Many data changes take place.
- The database is part of a 24 x 7 application.
- The database is part of a publish-and-subscribe or replication system—for example, log-shipping architecture

The full backup-and-restore model incorporates a higher frequency of data backup in a round-the-clock operation. Additionally, the system databases must be part of the backup strategy, since these contain data related to the user-defined databases and server configuration. The purpose of this is that you are able to restore your database up to the

moment of failure. The full backup strategy is more elaborate than that of the simple backup-and-restore model. You will use a mix of backup options, described in the following sections, to implement this model.

The restore part of this model is also complex. Not only do you need to restore more databases, but these restores must be done in different stages. For mission-critical databases, full backup is the recommended strategy. Microsoft SQL Server 2000 has a number of backup options that enable you to construct a fine-tuned backup strategy.

Database Backup Options

Saving the content of your database to another storage device can be done in a number of ways. If these options are used in the proper way, you can have an efficient backup strategy with minimal loss of data.

Complete Backup

A *complete database backup* writes all data in the database to the backup storage media. Depending on the size of the database, this can be a lengthy operation. Therefore, a complete backup should be done at a larger interval than the other backup options. This interval will increase with the size of the database. In fact, it has everything to do with the duration of a complete backup. For example, if the database is used only during office hours, you can start the backup during the evening hours; if the backup finishes before office hours commence the next day, it is sufficient to continue on this schedule. If the database is used and the content changed during a complete backup, these changes will not be part of the database, and you will need the transaction log to complement the backup.

The frequency of complete backups can depend on the time it takes to complete a backup and the size of the database. During the implementation of the backup strategy, you can determine the available backup time. When this time is exceeded, you need to lower the frequency of the complete backup. The other backup options must replace the complete backup. Besides the possibility of a full backup taking up too much time, the database can be so large that you run out of backup storage (a full tape) before the backup is complete. Both issues—increasing the backup performance and enlarging the backup storage capacity—can be tackled using multiple backup devices. SQL Server 2000 is able to do a parallel backup of a database to more than one backup device. Since all the dedicated backup devices are used simultaneously, the number of devices can roughly divide the total backup time. The total backup capacity is equal to the sum of the capacity of the available backup devices. Two important notes in this respect are:

- With the use of multiple backup devices, SQL Server does not allow you to mix SQL Server backup with other backups—for example, system backups. This is allowed when using one backup device, however.
- You can restore a backup made by multiple backup devices using one restore device.

Differential Backup

Between two complete backups, segments of data in the database, called *pages* , will be changed and new data will be written. Instead of making complete backups over short intervals, you can choose to back up only the changed pages—the differences between the complete backup and the current state of the database. This is called a *differential backup* and holds all changed pages since the last complete backup, even if another differential backup has been made. This type of backup enables you to roll back the database to a certain point in the past by restoring the complete backup and one appropriate differential backup.

In the case of a large database, you will keep making differential backups, up to the moment the backup size becomes too large or can no longer finish within a certain amount of time. For example, say that you have a 100GB database and the differential backup is 50GB, and it takes about 4 hours to complete. Both values could be the cue to make another complete backup. Especially in the case of a restore, you must restore 150GB for a 100GB database, costing valuable time.

Since the database will have changed since the last differential backup, you will also have to restore the transaction log after the differential backup to return the database to its most recent state.

Transaction Log

Every change made to the database is recorded in a *transaction log* in a way that it can be played back. This transaction log can be saved to backup storage. After each backup of the transaction log, it is truncated to the active portion, thereby keeping the transaction log small. The active portion consists of the incomplete transactions (see Figure 7.2). In case of a database failure, it is possible to back up the current transaction log, preventing the loss of committed transactions. The active portion with the incomplete transactions must be regarded as lost.

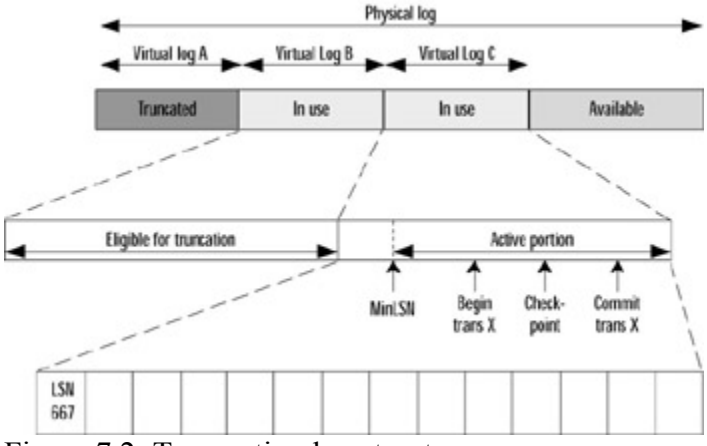


Figure 7.2: Transaction log structure.

You can increase the chance of saving the transaction log by placing it on a different physical disk from the one on which the database resides. The frequency of the transaction log backup must be higher than that of the differential backup. It is also prudent to back up and then truncate the transaction log following a complete or differential backup. Since the transaction log is also used in certain replication operations (for example, with log shipping), transaction logs must be synchronized.

Restoring a database backup and transaction logs can achieve a rebuild of the database just before the point of failure. The transaction log is the only file that can enable you to minimizing the loss of data. The complete and differential backup of the database contains only the active portion of the transaction log, meaning that the incomplete transactions become part of the backup.

The transaction log is a physical file, the size of which can be managed from within the Enterprise Manager. Within this file, SQL Server creates virtual log files. SQL Server manages the number and size of the virtual log files that reside in the transaction log file. The size is based on the number of transactions recorded. After a virtual log is full, it can be truncated. With a fixed-size transaction log, SQL Server uses it in a cyclic manner. This means that SQL Server starts logging at the beginning of the transaction, when it reaches the physical end of the file. The active portion of the transaction log can reside in a virtual log file that is not currently active. Only virtual log files that are not in use and are not part of the active log can be truncated.

Every new record that is inserted receives a unique identifier, called a *log sequence number (LSN).* A record can signal the beginning of a transaction, the commit of a transaction, an update within a transaction, or a checkpoint The active portion always starts with a new transaction and is identified by the minimum recovery LSN. The record at the start of the active portion of the log is identified by the *minimum recovery log sequence number (MinLSN).* Additionally, SQL Server flushes the database pages that contain modifications to disk. This action is recorded in the transaction log with checkpoints. The MinLSN is the minimum LSN of the following four conditions:

- The last checkpoint
- The start of the oldest active transaction
- The oldest modified database page
- The start of the oldest uncommitted replication transaction

This means that the active portion can span more virtual logs, limiting the possibility of backing up the transaction log and freeing log space.

File and Filegroup

The full database comprises a number of files on the storage disk. The actual database content can be spread over more than one file. A file must be placed in one *filegroup* . SQL Server 2000 has the option to back up the files and/or filegroup separately. For example, the database could exist in two files: the primary, called *primary.mdf,* and the

secondary, called *secondary.ndf*. All but the primary data file has the extension .ndf. If no additional filegroup is created, both will be part of the filegroup called *primary* .

You can devise a backup scheme that alternately backs up the data files every other night. This method can cut down on the time used to back up as well as the time involved in restoring a database. Since just one of the files could physically be damaged, you have to restore only that particular one.

In the case of a very large database, it could exist in a larger number of files, distributed over more than one filegroup. Backups can also be made based on filegroups. After restoring a file or filegroup, the transaction log backups have to be restored.

Backup Storage

In most cases, a backup file will ultimately be stored on removable media. The choice of backup storage is dependent on price, backup/restore speed, capacity, and the period of time you want or need to save the backups. Although in many cases a tape medium used, you can also use optical or magneto-optical disks or onsite or offsite disk storage—for example, backing up over a remote connection. The backup speed can be improved using multiple backup devices. You can also improve speed and reliability of the backup strategy by first backing up transaction logs and differential backups to disk, before transferring them to the final backup device. The local disk storage is often a faster medium than the other backup devices. This also implies that for a restore, you need to retrieve the backup files to disk first, before you can do the actual database restore.

Determining Storage Requirements

To be able to recover a database from a failure—or worse, a disaster—you must have all necessary data at hand. If all backup files are located on one tape, your restore strategy can run into dire straits if you receive errors reading the tape or the tape is damaged. The best way to back up the data is to use more than one backup set, enabling you to perform a restore even if one tape is unusable. Before you can determine your exact storage requirements, you need to answer the following questions:

- What is the current backup size of the database(s) you need to back up?
- What is growth rate of the database(s)?
- What are the size and the growth rate of the differential backups and transaction log backups?
- What is the retention period of complete and differential backups?
- Which backup files are stored online and which offline?
- What are the scheduling intervals of differential backups and transaction logs?
- How much time is available in which to make the backups?
- Which media rotation plan is used?

You should take some time to consider these points. The first question is easy to answer; the second and third are not. These are dependent on the application to which the

database is linked. The only way to approach this process is to start with a safe assumption and keep monitoring growth of the database and backups, then take action if these become significant in relation to the initial numbers.

The retention periods are related to the validity of the data and the way you should derive a media rotation plan. Suppose that every four weeks you make a complete backup; this would mean that up to day 27, differential backups must be made. If you wrap up the differentials, transaction log, and backup of the system databases weekly, the other differentials become invalid. Remember that differentials keep growing until the moment of the (monthly) complete backup, so to be safe, a differential, including the transaction log, must be saved to tape daily. During the day, you should save transaction logs to disk every hour (or half hour if the volume of changes is large) and make 6 or 8 hours of differential to disk (depending on the level of changes and the size of the differential). After the daily backup, the transaction log and differential backups can be deleted.

If the period in which the backups must be completed becomes too small, you must consider other schemes. Since you can back up the database online (with some restrictions; see "Backup Strategy Options" in this chapter), a backup should theoretically be successfully finished within 24 hours. However, if a differential backup is not finished within 6 hours and a complete backup not within 8 hours, you must implement a different approach for saving the database. You can consider parallel backup using more tape units, database replication, or a storage subsystem that has efficient ways of coping with a very large amount of data. To determine the size of the database, you should look at the "in use" size of the database files, not the "allocated" size on the storage device. The SQL Server 2000 backup utility writes only the used pages to the backup medium.

Backup-and-Restore Strategies in Replication and Publish-and-Subscribe Environments

The strategy used to back up a single database is critical, but when confronted with a database with content that is replicated, you need to consider even more issues. When you picture replication as a chain of content relationships, you have on one side the sender and on the other the receiver, or on both sides senders and receivers. Besides restoring the database, you can rebuild it. Deriving a fitting strategy that enables you to recover from a failure additionally forces you to get the relationship chain synchronized again.

The first step is the backup strategy. It is firmly recommended that the backup strategy for each database be the same as the one you would use for a database with no replication relationships. The best approach is to assume that you will not be able to recreate the database from a source other than the backup files. Be aware of the fact that the backup also saves information about the database's replicating state.

When you have the backup strategy in place, you can take a closer look at the way the restore and subsequent synchronization need to take place.

Since there are three types of replication and each type can have three types of databases, there can be at least nine different methods for restore and synchronization. The types of replication are transactional, snapshot, and merge. The latter is the most difficult to restore and synchronize, since it involves a two-way synchronization. The types of databases involved in replication are publisher, distributor, and subscriber. For this reason, you have at least nine combinations for which you need a restore strategy.

Before we get into these different strategies, the following is important to know: When you do a full restore of a database, the SQL Server checks the sysdatabases.category column of the master database to see if any replication settings in your database need to be preserved. For publishing databases, this column is used to indicate that replication is enabled. If you restored a replicated database to a different server or database, you would lose the replication information. If you also use a distribution database, you must perform a coordinated restore with the publisher database, preventing them from getting out of sync. The same is the case when you back up the databases of any publisher/distributor relationship. During the backup of a subscriber database (a transactional or snapshot replication), table Msreplication_subscriptions is also backed up. This table records the last replication transaction received. This information makes synchronization with the publisher or distributor possible.

In order to restore a replication environment, SQL Server provides a number of features that makes recovery more efficient:

- **Replication scripting** You can write or generate a script that can reestablish a replication relationship after you are forced to restore a publishing, distributing, or subscribing database.
- **Replication validation** This feature can perform an automatic resynchronization between a publisher and subscriber after the replication relationship is disrupted—for example, because one of the databases needed to be restored.
- **Snapshot file backup** The publicized replication transactions are stored in snapshot files on the distributor server and can also be backed up. If these files are large, it is faster to restore them instead of regenerating them, in case a distributor database must be restored.
- **Automatic synchronization of a publication** After a server resubscribes to a distributor, it can automatically synchronize using the snapshot files of a publication, without forcing the distributor to regenerate this file.
- **Retention period of snapshot files** If you increase the retention period of the snapshot files, you can use the automatic synchronization in more situations in which a resynchronization needs to take place. The drawback to this method is that you need additional disk space to store the snapshot files.

Now let's take a look at some strategies you can use to restore a database in a transactional or snapshot replication. It is assumed that the backup strategies for all databases involved are equal and take place in a coordinated fashion, enabling an adequate way of restoring replication relationships:

- Restore strategy for a *failed subscriber:*
    1. If a reliable, reasonable, up-to-date backup is available, restore this database. Validate the data of this restored database to the database to which it has subscribed. If thevalidation fails, the subscriber needs to reinitialize the database.
    2. If no valid database is available, you need to drop the subscription at the publisher/distributor end and recreate it at the subscriber.

- Restore strategy for a *failed publisher:*
    1. In the case of only subscribing databases, you must restore the publisher database to the latest state. The next step is running the Configure Publishing and Distribution Wizard to reenable replication.
    2. In a relationship with a distributor in which a coordinated backup between publisher and distributor is available, restore the publishing database. The next step is to restore the distributor's database and snapshot files. Then validate the data of the subscribing databases to the distributor database. If the validation fails, all subscribers need to reinitialize the database.

- Restore strategy for a *failed distributor:*
    1. If a coordinated backup between publisher and distributor is available, restore the publishing database. The next step is to restore the distributor's database and snapshot files. Then validate the data of the subscribing databases to the distributor database. If the validation fails, all subscribers need to reinitialize the database.

If none of these strategies works, you need to rebuild the whole replication topology. In this case, the availability of a replication script can be highly supportive in an efficient way. Be sure to disable or drop the replication configuration on all servers involved.



Backup Storage Media

If we consider backup storage media, the first option that comes to mind is the tape cartridge. This is still the cheapest solution that is also more robust than disks and is meant to be removable. Depending on the amount of data to be backed up and the speed at which this must take place, a number of brands and types of tape units are available. Within SQL Server 2000, you can distinguish three types of backup media:

- **Disk** SQL Server 2000 can back up a database to a file on a disk. This disk can be on the same server or on a different server, as long this disk is accessible through a drive or directory designation. Files created on disk will have the default name: <database name>_<date & time>.bak. It is possible to append backup files to existing ones on disk, but this is not the common practice.

- **Tape** A tape device is supported by Windows 2000 and directly connected to the server from which the backup will be made. Before SQL Server can back up the database to tape, it must first be formatted. More backup files can be contained on one tape cartridge. Subsequent backup files can be written to tape after the last backup file. Depending on the configuration, all backup files on a tape can be overwritten when backing up a file to tape. SQL Server 2000 will start saving the file at the beginning of the tape.
- **Named pipe** This is an interprocess communication (IPC) mechanism that lets SQL Server 2000 access shared network resources. A named pipe is a shared memory resource file that is used by the initiator (the client) to write to and by the receiver (the server) to read from. The receiver can be a device agent, enabling you to access a remote storage device. Named pipes always have the following default name construction: \\<Server_Name>\pipe\<pipe_name>.

The most popular tape cartridge types are *digital data storage* (*DDS* , also known as *DAT-DDS,* for *digital audio tape-digital data storage*) and *digital linear tape (DLT).* Depending on the amount of data you have to save, you can choose the most appropriate tape technology (see Table 7.1). Additionally, by choosing a tape unit that incorporates an autoloader (or stacker), you can extend the backup capacity. The number of cartridges that can be held by an autoloader is limited to about seven. If your data-base grows beyond the total storage capacity provided by an autoloader, you must consider other options—multiple auto-loaders or even tape library systems.

Table 7.1: Capacities of Various Tape Technologies

Storage Type	**Maximum Storage Capacity**
DDS	2GB uncompressed
DDS-2	8GB compressed
DDS-3	24GB compressed
DDS-4	40GB compressed
DLT III	20GB compressed
DLT III XT	30GB compressed
DLT IV	70GB compressed

Note It is important that you record the date of the first use of the cartridge. Tape has a limited lifetime due to wear caused by the direct contact between tape and read/write heads. For example, a DDS tape is “guaranteed” for 2,000 read/write passes and an expected life of 10 years. Remember that when a tape is not used, the magnetic imprint gradually loses its strength, hence losing the data stored on it. Therefore, it is a good policy to retire tapes after a certain amount of time. A rule of thumb is to retire tapes that are used for daily backups after one year. Tapes for weekly backups should be retired after two years.

The heads of the tape units should be cleaned on a regular basis—at least once a month. Even with tape units that signal a cleaning operation, proactive cleaning

reduces the chance of failures.

Never let tapes lie around the computer room or your desk. You should store them in a data fault immediately after use. If disaster strikes, you want to be sure you can retrieve the tapes out of the rubble. *Never* try to save money on a data vault. It is important that the vault be able to hold all your data cartridges on site. You should examine independent test reports of data vaults (such as those from Underwriters, at www.ul.com). A data vault has a few specifications that must be considered at the time of selection. Important is the temperature it can resist and the duration it can resist this temperature. In addition, impact resistance is also a significant specification. The vault needs to be able to resist impacts from falling down or debris falling on it. Always place the vault as far away from the computer room as possible. Then, in case of a local disaster in the computer room or in an adjacent room or hallway, you still have access to the backups.

Regular or heavily used tapes, such as those used daily to store backups of multiple databases, can become victims of tape tension problems. The tape can become subject to irregular tape tensions that can ultimately lead to read and/or write errors. To prevent this problem, re-tension your tapes at least twice a year. Most advanced backup applications provide this functionality.

Media Sets, Media Families, and Multiple Drives

Surrounding the backup and restore of SQL Server, a number of terms that are used can become confusing, especially if you need to understand larger backup constructions. The terms are summarized and defined in Table 7.2 .

Table 7.2: Media-Related Terms

Term	Definition
Backup file	Holds a complete, differential, transaction log or file(group) backup.
Backup set	Holds all the backup files of a single backup operation.
Backup media	A disk, tape, or named pipe that is used to physically store a backup set.
Backup device	A tape, disk file, or named pipe used in a backup or restore operation.
Media set	All media involved in a backup operation; a media set can exist in one or more media families.
Media header	A block of information about the backup media, written at the beginning of the media.

Table 7.2: Media-Related Terms

Term	Definition
Media name	The name describing the complete media set; the name appears in the media header and can identify the backup media that belong to the same media set for the entire backup media set.
Media family	All backup media that belong together and are written to by the same backup device. The first backup media is called the initial media, and the following backup media are called continuation media. Although a single backup media is a media family, after the backup files grow to a certain size, you need more backup media to store the backup files. A media family can grow, without limits. Notice that continuation media can exist only for tape devices.

If a media set exists in more than one media family with one or more backup media, more backup devices must be involved. This construction will be used to improve the backup performance when very large amounts of data must be backed up. Since the SQL Server backup process will perform a parallel backup to all available devices, keeping backup media neatly organized is very important.

As the number of devices increases, there is a point at which manually handling the tapes is not recommended. Instead, it is recommended that you start using a *tape library unit (TLU)* . SQL Server and the TLU driver software will take over the management of the backup media. Once you start making backups over multiple drives, you must continue doing so. The SQL Server backup process will continue even if one of the devices runs out of tape space in its media family. However, there is a situation that will stop the backup process; this has to do with the way the backup tries to keep track of the distributed backup: by building in specific synchronization points during the backup. If such a synchronization point occurs while one of the backup devices is awaiting a new backup media, the backup process will stop altogether until that backup device is fed a new media.

The use of a TLU has many advantages but also one disadvantage, which is the part of the backup strategy in storing the backup media in safe storage (onsite or offsite), especially since this backup process will become a continuous process. Therefore, it is important to replace the whole media set directly after the backup is completed. *Always* keep the media set together! Losing one tape will invalidate the whole set. Construct a strict regime in the way the tapes must be handled.

A solution involving multiple drives is highly recommended in situations in which you have a mission-critical, fast-growing database to which users need 24 x 7 access. Using two tape units with an autoloader, you will be able to cut the backup time in half, and you'll also be able to make more backups daily.

Media Rotation

As mentioned previously, you should use different levels of backup: monthly, weekly, daily, and hourly. The first three are normally written to tape and are referred to as a *grandfather-father-son (GFS)* rotation scheme. This method requires more tape cartridges in the media pool. After a year, you can restart the backup cycle. A good practice is to make the last tape of the year a "year tape," containing a full backup of all related files and databases, and retire it.

For the monthly backup, we use an interval of four weeks (28 days). You need 13 tapes to cover a complete year. The weekly backups cover three tapes, and the fourth one is the monthly one. The daily backups are done for six days, and the seventh day is the weekly backup. Therefore, you need six tapes for daily back-ups. In total, you will have an initial backup set of 22 tapes. If the backup data do not fit on one tape, you'll have to add tapes to the media set.

Secure Offsite Storage

Although it is efficient to have the tapes at hand, in case of a disaster, you'll wish that you had the tapes stored somewhere else. There are companies that sell "disaster services" providing safe, secure offsite storage of your tapes. The place in which they store the tapes should be at least a one-hour drive and at most a four-hour drive from your facility. It should not be too close, so that your data will be safely situated outside a large potential disaster area. It shouldn't be too far, however, since valuable time will be lost in retrieving the tapes if you need them. Companies that manage extremely large amounts of data, such as banks or insurance companies, send their tapes to storage on a daily basis, through a qualified courier service. The data on the tapes represent immense sums of money to the safe storage facility.

For smaller companies, one weekly ride to the storage facility can be enough. A weekly swap of six daily and one weekly tapes seems a good thing to do. This means that you need 12 cartridges for the daily backup. One set is in the secure offsite safe storage, and the other one is in your onsite data vault.

Sample Backup Scheme

Now that we've discussed many issues surrounding backup strategies, it is time to put them together in one backup plan. Let's assume that the Northwind database is our mission-critical database that we need to back up in a reliable and save way. For demonstration purposes, the Northwind database is made up of two files residing on different drives for performance reasons. The transaction log file resides on a third drive,

primarily for recovery reasons, but it also can improve performance. The backup scheme will look like this:

- Every four weeks (monthly), on a Saturday, a backup is made to tape of the following:
    - Complete Northwind database
    - Transaction log
    - Complete master database
    - Complete msdb database
- Every week (weekly), on a Saturday, a backup is made to tape (except when a monthly backup is run) of the following:
    - Complete Northwind database
    - Transaction log
    - Complete master database
    - Complete msdb database
- Every day starting at midnight, except when a weekly backup is run, a backup is made to tape of the following:
    - Differential Northwind database
    - Transaction log
    - Complete master database
    - Differential msdb database
- Every 6 hours (6:00 am , 12:00 am , and 6:00 pm), a backup is made on disk (on another server) of the following:
    - Differential Northwind database (differentials older than 24 hours are removed)
    - Transaction log (transaction logs older than 24 hours are removed)
- Every hour, a backup is made on disk (on another server) of the following:
    - Transaction log
- Before a differential or complete backup is made, an integrity check on the database must be done
- Before a monthly backup, the database must be optimized; a better strategy would be to make a complete backup, then optimize and create a complete backup again
- Set for each backup the appropriate retention/expiration dates: 391 days for a monthly backup ((13+1)*28)-1); 27 days for a weekly backup; 13 days for a daily backup (2*7-1); and remember that we run with two sets of daily tapes, one onsite and one offsite
- The transaction log and differential backups that are made on disk storage on a separate server are also backed up through the regular system backup application
- An operational procedure describes how tape changes take place and how weekly tape sets are exchanged with the secure offsite storage

It is good practice to use a logbook in which you record the performed backups. Additional information should include:

- Whether or not all backups succeeded

- How much space is left on the tape and how much storage the backups need (in case the backup tends to take more than one tape cartridge)
- Possible errors and/or warnings in the SQL Server log file (located in the default directory \\<server_name>\mssql2000\log) or in the application or system log (accessible through the Event Viewer)
- Dates on which the read/write heads of the tape unit are cleaned
- Signoff from the administrator on duty

The use of this type of logbook obligates you to perform the checks and enable other administrators to substitute for the responsible administrator.

Backing Up the Master and msdb Databases

In the backup plan, the master and msdb databases are backed up. Since these databases hold important operational data related to our own business database(s), there is every good reason to back up these databases, too.

The master database holds information related to the following:

- User accounts
- Remote servers
- Configurable environment variables
- Available databases
- Storage space allocated to each database

The msdb database holds information on:

- Online backup and restore history
- Scheduling tasks
- Replication
- DTS packages

Without backing up these databases, restoring your database can become a very tedious task.

Creating a Recovery Strategy

After deciding on a backup strategy, you can derive a recovery strategy. Remember that the recovery strategy is as good as the data at hand. A recovery strategy must be able to execute recovery from disasters ranging from a disk failure to a flood. The intention of the recovery strategy is to rebuild a working situation in which the database is online again in the quickest possible way and losing as little data as possible. We make a distinction between local failures, as severe as losing a whole database server, and disasters, which can range from a small a small fire in the computer room that wipes out all servers and storage to a large disaster that turns the whole building into rubble.

The first step in the recovery strategy is the creation of a recovery procedure. This procedure should ensure that a recovery (or restore) is always done in the same way and no steps are missed. From the moment you detect a failure that denies access to the database, you should perform the following steps:

1. Determine if, despite the failure, the database is still operational and not corrupted (the failure can be network related).
2. If the database server is still accessible through the Enterprise Manager, make a final backup of the transaction log, saving as much data as possible.
3. If the database is no longer functioning or not functioning properly, bring the database server and all related processes down.
4. If a restore of the complete database must take place, the last weekly backup must be retrieved from the secure offsite storage location.
5. Determine what part failed and analyze the possible or probable root cause.
6. If the failure can be quickly repaired, do it. If the repair and restoration of the server, storage, or another part of the system takes more time, move the databases and SQL Server 2000 to another server.
7. Collect from the onsite data vault all tapes that might be needed during the restore.
8. After the database server is operational, the restore can take place again.
9. Restore the databases. After each restore, check the integrity of the database as follows:
    - The complete backup of the master and msdb databases.
    - The complete backup of the business database.
    - The last differential of the master and msdb databases. These reside on the last daily backup that was successful.
    - The last differential of the business database. This can reside online on the server disk storage to which these backups are made. If these are not available, the one on the last successful daily backup should be used, unless a more recent differential is available on the regular system backup of that server.
    - Apply all available transaction log backups. Only the transaction log backups made after the differential backup was made are valid. These should reside on the server disk storage to which the backups were made, starting from the time the restored differential backup was made.
10. After the database is online again, one or two experienced staff members must perform some tests to confirm that applications and databases are also functioning at content level.
11. Make a complete backup of all related databases and save or store them at the offsite storage location. In addition, return the tapes that were retrieved from the offsite storage location.

In case of a disaster, relocation must take place. The database recovery strategy has become part of the execution of your contingency plan. If you have ever run into this kind of situation, you know the importance of up-to-date documentation. Rebuilding the server farm on different hardware could involve a lot of manual labor. Relying on

backups is not good enough. These rebuilding operations always incorporate challenges for system administrators. A great deal of valuable time will be lost if you don't have the proper documentation. Because a disaster will result in loss of data, it is prudent to determine up front if, and in what way, content can be manually rebuilt.

Backup and Restore Tools and Techniques

Until now, we've discussed backup and restore strategies in general terms, since they apply to every database environment. However, it is now time to take a look the tools Microsoft SQL Server 2000 incorporates to help you implement your backup strategy. Remember that backup strategies should be automated, and restores must be done manually. The backup tools within SQL Server 2000 include the Create Database Backup Wizard, the Database Maintenance Plan Wizard, and Transact-SQL (T-SQL).

The Create Database Backup Wizard

The Create Database Backup Wizard (see Figure 7.3) enables you to construct separate backup tasks that can be scheduled or run manually. This wizard supports the backup of complete databases, differential databases, and transaction logs. If you want to back up based on files and filegroups, you must use T-SQL commands (see the "Transact-SQL" section that follows) and run these on designated times as part of a maintenance task.



Figure 7.3: The welcome screen of the Create Database Backup Wizard.

The Create Database Backup Wizard will lead you through the following steps:

1. Select the appropriate database.
2. Choose a name and description for the backup task.
3. Select the type of backup (complete, differential, or transaction log).
4. Select the backup device.
5. Select media set and backup lifetime.
6. Create a schedule.

Once you have completed these steps, the wizard will first give a summary of the backup task to be created. If the summary is correct, you can finish it; then the backup task will appear in the Jobs list of the SQL Server Enterprise Manager (see Figure 7.4). The list of jobs can hold all backup tasks for all databases. Every one of them can be scheduled or manually run. The wizard is a quick and effective way to create your backup tasks,

although you can also create them manually by choosing Management | SQL Server Agent | Jobs in the Explorer view of the SQL Server Enterprise Manager. Because the jobs have no relationships, it can be difficult to oversee how these jobs are sequenced in time. By giving the job a name or description that consists of the database and scheduling interval, you will be able to group backup tasks of each database together and directly see the intervals at which they are scheduled.

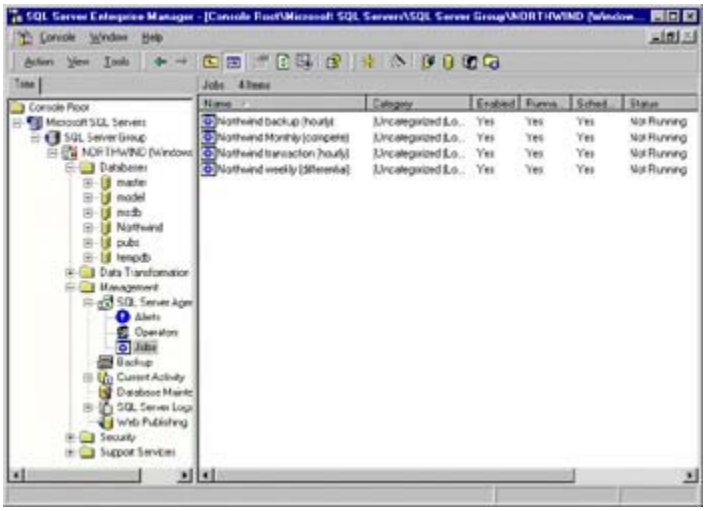


Figure 7.4: Backup jobs available through the SQL Server Enterprise Manager.

The Database Maintenance Plan Wizard

It is often desirable to group a number of tasks into a single job. For example, after each complete or differential backup, you might want to run a backup of the transaction log and, before starting the backup, do a integrity check of the database to ensure that the database backup is not corrupted. In this case, you can create maintenance plans manually or use the Database Maintenance Plan Wizard (see Figure 7.5). A major advantage of this wizard is that not only can you group more maintenance tasks in one job, you can also execute this same plan on more than one database. The more experienced database manager can construct maintenance plans using the command-line utility sqlmaint. With the Database Maintenance Plan Wizard, though, you are not able to implement backups based on files and/or filegroups.

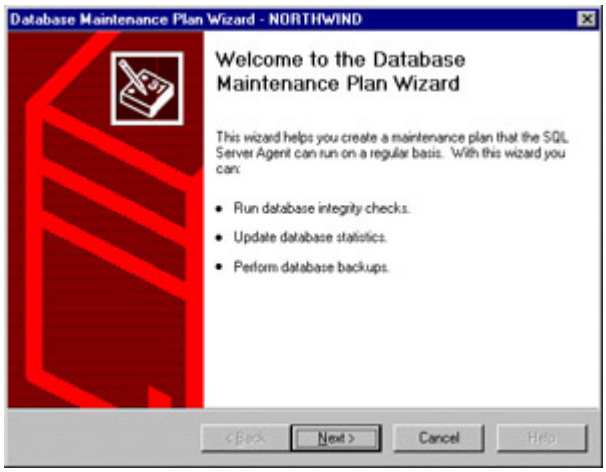
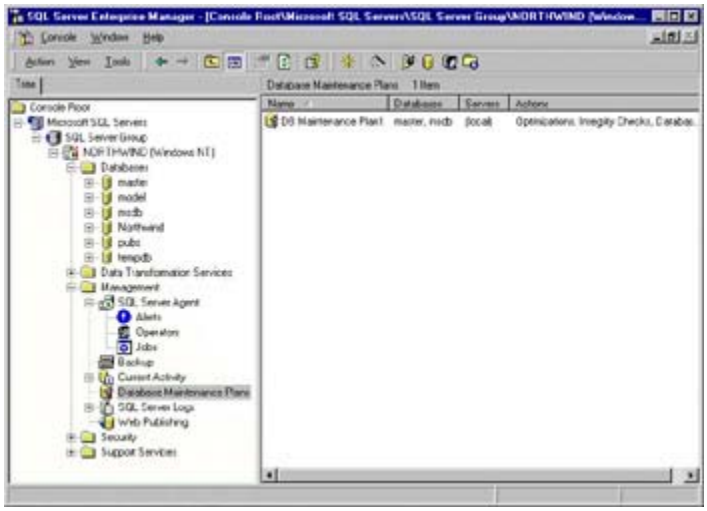


Figure 7.5: The welcome screen of the Database Maintenance Plan Wizard.

The Database Maintenance Plan Wizard will lead you through the following steps:

1. Select databases.
2. Choose the data optimization information options.
3. Choose the database integrity check options.
4. Specify the database backup plan.
5. Specify the transaction log backup plan.
6. Select the report options.

On completion of these steps, the wizard will first give a summary of the maintenance plan to be created. If the summary is correct, you can finish it, and then the maintenance plan job will appear in the Database Maintenance Plans list of the SQL Server Enterprise Manager (see Figure 7.6). The list of jobs can hold all backup tasks for all databases. Every one of them can be scheduled or manually run. (The latter is also true even if the task is scheduled.)


Figure 7.6: Maintenance plans available through the SQL Server Enterprise Manager.

Transact-SQL

T-SQL is an extension of the standard SQL that enables you to build scripts that directly interact with SQL Server. Instead of using the Enterprise Manager, the Create Database Backup Wizard, and the Data-base Maintenance Plan Wizard, you can write your own maintenance scripts. Under the hood, the jobs and tasks you create are recorded in the master database and run using T-SQL. Although you need to have some SQL programming experience to use T-SQL, the big advantage is that your can make tailor-made backup and restore scripts.

T-SQL incorporates the commands BACKUP and RESTORE. Additionally, SQL Server 2000 has stored procedures that enable you to build your own backup and restore jobs. You can use the Create Job Wizard to automate the execution of the T-SQL scripts.

Backing Up SQL Server Databases

Now it is time to put things to work and see how you can use the two wizards and T-SQL to your benefit in implementing a backup strategy. As an example, we'll use the backup strategy described in the "Sample Backup Scheme" section on the Northwind database. In order to get the distributed effect of storage, we'll use the drive and file assignments shown in Table 7.3 .

Table 7.3: Example File and Drive Assignments for the Northwind Backup Scheme

Purpose	**File**	**Drive**
Primary database file	Northwind.mdf	G:\Mssql\Data
Second (user-defined) database file	Northwind2.ndf	F:\Mssql\Data
Transaction log file	Northwind.ldf	D:\Mssql\log
Backup storage for transaction log	*.bak	H:\Mssql\backup and differential backup files

Both the Create Database Backup Wizard and the Database Mainte-nance Plan Wizard can be accessed through the Tools | Wizards menu option. This brings up a dialog box with an expanding selection tree (see Figure 7.7). Both wizards are located under the Management option.

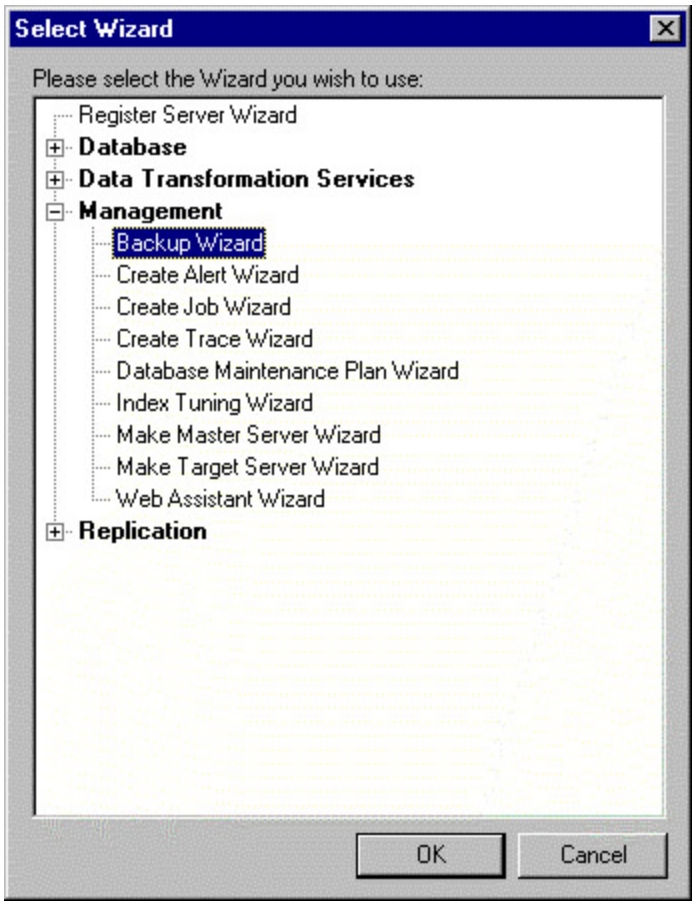


Figure 7.7: Wizard selection.

Performing a Database Backup

To show the use of the Backup Wizard, we'll use it to create a backup job for the monthly complete backup of the Northwind database. After you start the wizard, the opening welcome screen (refer back to Figure 7.3) is shown. Select the Next button, and you are asked to select the database for which you want to create the backup job (see Figure 7.8). By default, the master database is the selected database; make sure that for creating the sample monthly backup, the Northwind database is selected.

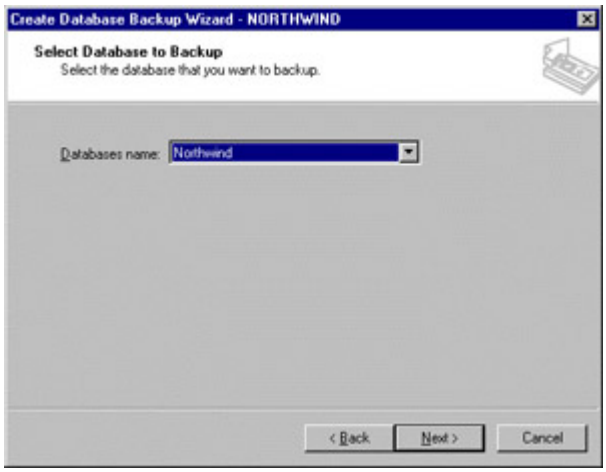


Figure 7.8: The Select Database to Backup dialog box.

In the next step, you will be asked to enter a name and description for the backup job. The name will be shown in the job list (refer back to Figure 7.4), so use a name that describes the job well; otherwise, you will need the job properties to find out what the job is supposed to do. Next you have to select one of the following three types of backup you want to perform:

- **Database backup** Back up the entire database (referred to as Database–Complete). This option is selected by default.
- **Differential database** Back up only new and changed data.
- **Transaction log** Back up the record of all the changes made to the database (since the last complete or differential backup).

For our monthly backup job of the Northwind database:

1. Select the Northwind database.
2. Press the Next button.
3. Enter **Northwind Monthly (complete)** for the name of the job.
4. Leave the description empty.
5. Press the Next button.

If you have selected the appropriate backup type, you must select the backup device to which you want to save the backup file (see Figure 7.9). You need to select one of three device types, of which the first valid one is the default:

- **Tape** This must be one of the database server-connected tape devices.
- **File** A filename on one of the disk drives accessible from the database server. Use the "…" button to the right to browse to the desired drive/ directory and select an existing file or create a new one. The default name is the name of the backup job with the extension .bak; the default directory is BACKUP, placed in the installation directory of SQL Server.

- **Backup device** This must be a device that is accessed through a named pipe. These named pipes do not have to exist before you use the Create Database Backup Wizard. By selecting <new backup device>, you can create a named pipe.



Figure 7.9: Select Backup Destination and Action dialog box.

The second part of the dialog box can be used to select some properties related to the backup device:

1. Select "Append to the backup media" or "Overwrite the backup media." The latter works only if the retention date of the backup file is expired; otherwise, the backup job cannot proceed. In the next step you can set the expiration date of the backup file. The first option is selected as the default value, since this prevents you from overwriting a backup file by default.
2. "Eject tape after backup" is valid only for tape devices.
3. You should select the "After completing the backup, read and verify the integrity of the backup" option by default. However, if the complete or differential backup is very large, this option can extend the backup period significantly. As long as you find that acceptable, leave this option checked.

If you have selected "Overwrite the backup media," you will be prompted with an Initialize Media dialog box that enables you to reinitialize the backup medium and give it a new name. If you give the backup file an expiration date that lies in the future, the overwrite option will not work and will result in a failed execution of the backup job.

For our monthly backup job for the Northwind database:

1. Select a tape device.
2. Select "Overwrite the backup media."
3. Check the Eject tape option.
4. Check the "After completing the backup …" option.
5. Press the Next button.
6. Check "Initialize and label media."
7. Enter **Northwind_complete** for the media set name.

8. Leave the media set description empty.
9. Press the Next button.

Now you are presented with a dialog box that enables you to enter expiration date and scheduling information (see Figure 7.10):



Figure 7.10: The Backup Verification and Scheduling dialog box.

1. If you selected “Append to media” in a previous dialog box, the “Checking media set” option is disabled.
2. If you selected “Overwrite media” and checked “Initialize and label media” in the previous dialog box, the “Checking media set” option is also disabled. The media set name you previously entered is used. However, if you left those options blank in the previous dialog box, it will now also be blank here, without the possibility to change it. Use the Previous button to change the option, if you desire.
3. If you selected “Overwrite media” but did *not* check “Initialize and label media,” the “Checking media set” option is enabled and can be checked. If you do this, you can also enter a media set name.
4. If you selected “Overwrite media” in an earlier dialog box, you are able to enter an expiration date. You have to check the option “Backup set will expire” to activate it. Now you have to fill in a duration. This can be done in two ways: first, by selecting After (also referred to as retaining days) and giving the number of days the media is not allowed to be overwritten. Second, choose a fixed date by selecting On (also referred to as expiration date), followed by a date somewhere in the future. The After option with the value of one day is the default.
5. The last section of the dialog boxes enables you to schedule the job. If you do not check Schedule, the backup is executed once after completing the wizard and will not become part of the job list.
6. After you check the Schedule option, the Change button is enabled, giving you the opportunity to schedule the backup job at the desired time.

For our monthly backup job of the Northwind database:

1. Check the “Backup set will expire” option.

2. Select the After option.
3. Enter **391** in the days field. (This number is explained in the section "Sample Backup Scheme.")
4. Check the Schedule option.
5. Press the Change button.

For a backup to be performed somewhere in the future or on regular intervals, you need to schedule it. By selecting the Change button, you enter the Edit Schedule dialog box (see Figure 7.11). A number of options are presented:

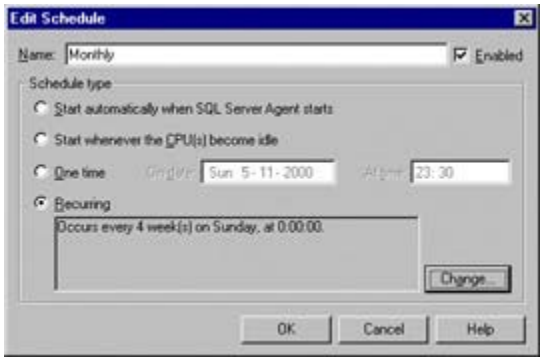


Figure 7.11: The Edit Schedule dialog box.

1. The Name field is used to give the schedule a distinguishing name. Although this wizard abstracts many underlying details, the wizard needs to create a schedule in the msdb database so that your job can enter the scheduling list.
2. The Enabled field is checked by default. If you uncheck it, the schedule is entered in the list but will not be executed at the appropriate time.
3. SQL Server supports four schedule types from which you need to select. By default, the Recurring type is selected with the value "Occurs every 1 week(s) on Sunday, at 0:00:00." This means that the backup job is executed every Sunday at midnight. The other schedule types are:
    - **Start automatically when the SQL Server Agent starts** This option is not really meant for backup purposes, but for other maintenance issues. The agents start on distinct moments, that is on the startup of the server or manual by a administrator, after the SQL Server (and agent) has temporarily been shutdown, for example after a restore.
    - **Start whenever the CPU(s) become idle** This is also a schedule type used for continued monitoring or maintenance jobs. This option schedules a job during the times when the database server has no workload. Since you do not know up front when the CPU will become idle, creating backups based on this schedule type is very unreliable.
    - **One time** This option includes a date and time at which you can schedule a backup somewhere in the future. This can be useful if you are planning major maintenance or an upgrade on a database. You can schedule a complete backup on Sunday evening so that you can start the maintenance/upgrade right away Monday morning, without losing time over the (very important) creation of a complete and reliable backup. By

the way, for this kind of backup, you should definitely check the option "After completing the backup, read and verify the integrity of the backup" (refer back to Figure 7.9).

For our monthly backup job of the Northwind database:

1. Enter the name **Monthly** .
2. Keep Enabled checked.
3. Select the Recurring schedule type (if you have not already done so).
4. Press the Change button.

By selecting the Change button, you enter the Edit Recurring Job Schedule dialog box (see Figure 7.12). The fact that the Job name has the value "(New Job)" is correct. This value will change only after you complete the Create Data-base Backup Wizard process and the job is entered in the msdb database. If you edit the job later on, you will see that the correct name (in this case, "Monthly") shows up after "Job name" in the dialog box.

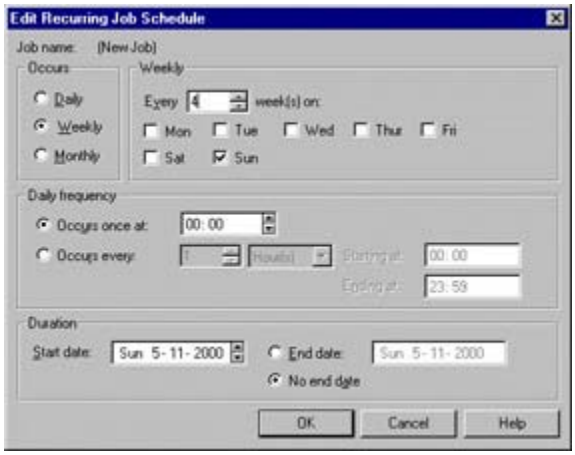


Figure 7.12: The Edit Recurring Job Schedule dialog box.

The Edit Recurring Job Schedule dialog box has three parts:

- The Occurs option lets you select the execution interval between two backup jobs. You have three options:
  - **Daily** By selecting this occurrence, you can enter the daily interval for which you want the job to be scheduled. For example, you could choose every day, but you could also choose once every three days;
  - **Weekly** For the weekly occurrence, you can enter the number of weeks that should pass between two consecutive scheduling moments of this job. Additionally, you can select the preferred day(s) for the job—for example, every other week, on Sunday and Wednesday. Weekly with a one-week interval is the default value.
  - **Monthly** By selecting this occurrence, you can enter the number of months that should lie between two consecutive scheduling moments of the job. Additionally, you must choose between monthly scheduling

moments. This choice can be a particular day, such as the second Monday of every month, or a particular point of the month, say, the seventh day of every month. Remember if you use the latter option, the job will not be scheduled, because the month has fewer days than the one you entered.

- The second option is used to define the Daily Frequency. This can be the time of the day, such as 00:00 hours, or a repeated scheduling during the day, such as every six hours. Even when you have a job that is run weekly, you can still schedule it to run a number of times during that day.
- The last part is used to define the validity of the job schedule. This option is used to decide if the job is still valid to be executed. The job will be scheduled after the first valid moment after the start date, and it will stop being scheduled after the end date. If no end date is selected, the job will be scheduled "forever."

For our monthly backup of the Northwind database:

1. Select Weekly (if it is not already selected). Although this job is meant to be scheduled monthly, it was chosen to run every four weeks. That is why Weekly is selected. If the choice for every first Sunday of the month had been made, Monthly would be selected.
2. Enter **4** in the "Every …week(s) on" field.
3. Check Sunday as the day of the week and leave the other days unchecked.
4. Select "Occurs once at."
5. Enter the time 00:00.
6. Leave the Start date on the default value.
7. Leave the "No end date" selected.
8. Press the OK button in the Edit Recurring Job Schedule dialog box.
9. Press the OK button in the Edit Job Schedule dialog box.
10. Press the Next button in the Backup Verification and Scheduling dialog boxes.

The wizard is about to create the backup job; it gives a summary (see Figure 7.13) of the values it is about to use for this creation. You are still able to make changes until you select the Finish button.

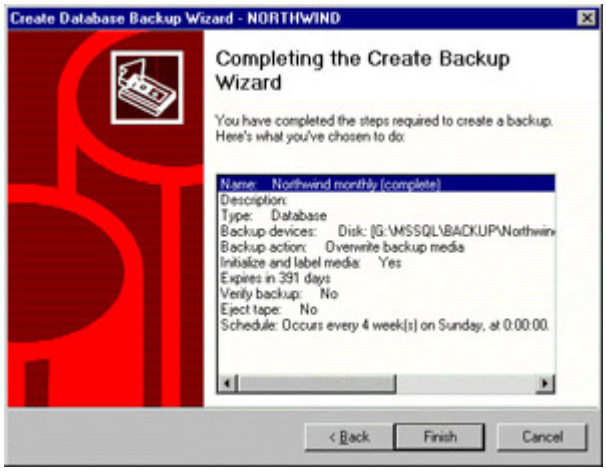


Figure 7.13: Completing the Create Database Backup Wizard.

To complete our monthly backup job of the Northwind database, press the Finish button.

In the same way, you can create the weekly, daily, and hourly backups. The summary of weekly differential backups is represented in Figure 7.14 .

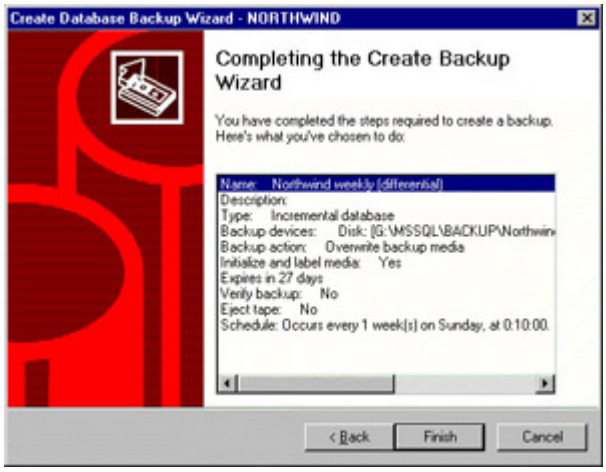


Figure 7.14: A summary of weekly differential backups.

Notice the choice to schedule the job a little past midnight. It is done on purpose and forces the job to be aborted on the same day the monthly backup is running. Since the monthly backup starts at 00:00, it will be activated before the weekly job schedule, assuming that the backup will take longer than 10 minutes to finish.

As stated previously, the Create Database Backup Wizard will turn the values you entered into a T-SQL script that is part of a job. This is important to understand if you want to edit a backup job—or any job, for that matter. The list of backup jobs is located in <Your_Database_Server> | Management | SQL Agent | Jobs in the Enterprise Manager (refer back to Figure 7.5). By double-clicking a job in the list, you'll cause the Properties dialog box to appear (see Figure 7.15). The tabs General, Steps, and Schedules hold the data that relate to the data you entered through the Create Database Backup Wizard.

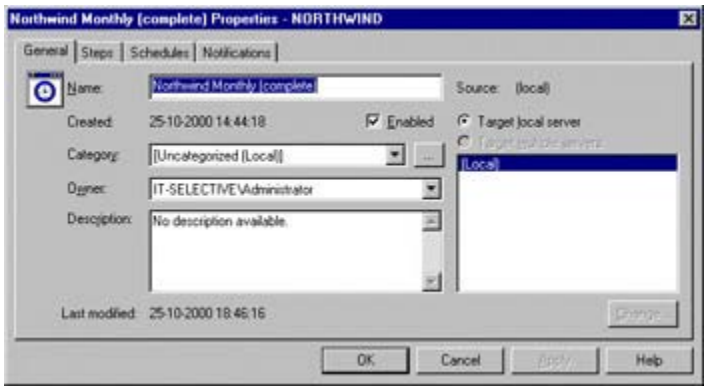


Figure 7.15: The Job Properties dialog box.

By completing the following steps, you will open a dialog box (see Figure 7.16) that shows that actual backup command:

1. Select the Steps tab.
2. Select the line with the step name Step 1.
3. Press the Edit button.

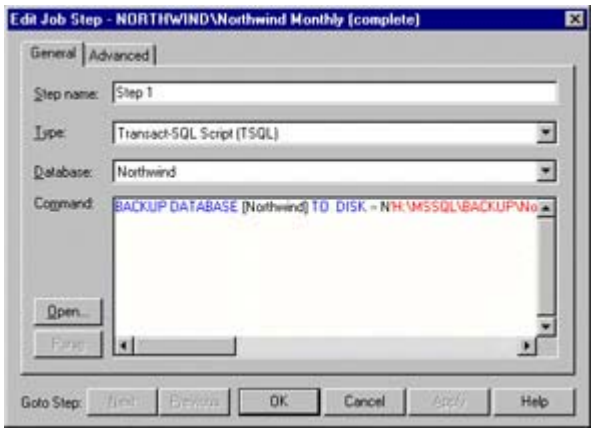


Figure 7.16: The Edit Job Step dialog box.

The T-SQL command that is entered in the text field Command looks like the following. For an explanation of the arguments, see Table 7.4 .

```
BACKUP DATABASE [Northwind]
    TO DISK = 'H:\MSSQL\BACKUP\Northwind Monthly (complete).BAK'
    WITH
            INIT,
            NOUNLOAD,
            RETAINDAYS = 391,
            NAME = 'Northwind Monthly (complete)',



            NOSKIP,
            STATS = 10,
            FORMAT
MEDIANAME = 'Northwind_monthly'
```

The weekly backup job has the following T-SQL command:

```
BACKUP DATABASE [Northwind]
    TO DISK = 'H:\MSSQL\BACKUP\Northwind weekly (differential).BAK'
    WITH
            INIT,
            NOUNLOAD,
            RETAINDAYS = 27,
            DIFFERENTIAL,
            NAME = 'Northwind weekly (differential)',
            NOSKIP,
```

```
            STATS = 10,
            FORMAT,
MEDIANAME = 'Northwind_Weekly'
```

The backup of the second database file (Northwind2.ndf) for the monthly complete backup has the following T-SQL command:

```
BACKUP DATABASE [Northwind]
   FILE = 'Northwind2.ndf'
   TO DISK = 'H:\MSSQL\BACKUP\Northwind2 Monthly (complete).BAK'
   WITH
          INIT,



          NOUNLOAD,
          RETAINDAYS = 391,
          NAME = 'Northwind2 Monthly (complete)',
          NOSKIP,
          STATS = 10,
          FORMAT
MEDIANAME = 'Northwind_monthly'
```

The hourly backup of the transaction log is turned to work with the following T-SQL command:

```
BACKUP LOG [Northwind]
    TO DISK = 'D:\MSSQL\BACKUP\Northwind hourly (transaction).BAK'
    WITH
          INIT,
          NOUNLOAD,
          NAME = 'Northwind hourly(transaction)',
          NOSKIP,
          STATS = 10,
FORMAT
```

Table 7.4: Explanation of the BACKUP Commands

Argument	**Explanation**
BACKUP DATABASE	If the BACKUP command is followed by DATABASE, the command needs to make a complete or differential backup of the database.
BACKUP LOG	If the BACKUP command is followed by LOG, only the transaction log will be backed up. The backup starts from the point at which the last transaction log backed up to the active portion of the log file (assuming that the previous backup was successful). After the backup, the transaction log is truncated up to the active portion.
FILE	This argument specifies the name of the database file you want to back up.

Table 7.4: Explanation of the BACKUP Commands

Argument	**Explanation**
INIT	If this argument is used, all the data that are contained on the specified device are overwritten. However, the existing media header is preserved. The INIT arguments fails if: • None of the backup sets on the media has expired. This is controlled by the arguments EXPIREDATE and RETAINDAYS.
INIT	• The value of the NAME argument is not the same as the name on the backup media. The working of INIT is influenced by the presence of the argument SKIP. If SKIP is used as an argument, the validation of EXPIREDATE and RETAINDAYS is omitted.
NOUNLOAD	This argument prevents the tape to be ejected from the tape unit. NOUNLOAD stays active for all subsequent BACKUP commands, until a BACKUP command uses the UNLOAD argument.
RETAINDAYS	This argument specifies the total number of days, starting from the day the backup is made; a media set can be overwritten. This option cannot be used if the device being written to is defined as a named pipe. In addition, you need to use the INIT argument for RETAINDAYS to be used. Moreover, if SKIP is used, this argument is ignored.
DIFFERENTIAL	This argument specifies that a differential backup must be made.
NAME	If the backup set needs to be named, this argument must be used. This is optional and left blank if not specified. With a blank name, the overwrite check of INIT is bypassed.
NOSKIP	When this argument is used, the checks specified under INIT are performed.
STATS	With this argument, you are notified of the progress of the backup. The value specified is the percentage of the steps in which this progress is reported. The default value is 10, meaning that after every 10 percent, the progress is updated.
FORMAT	This argument reformats the complete media set, making the current content invalid (even if a password is set on the media). A new media header is created. Be careful using this command. If, for example, the media is one of a backup set of more media, the rest also becomes invalid, since you broke the sequence of the set.

Table 7.4: Explanation of the BACKUP Commands	
Argument	**Explanation**
MEDIANAME	This argument carries the name of the media. If specified, it must also match the media name already present on the media, or the backup will fail. If not specified, this check will be omitted. The same goes if the SKIP argument is used.

Backing Up System Databases

To show the use of the Database Maintenance Plan Wizard, let's create a maintenance plan job for the monthly complete backup of the system databases. After you start the wizard, you'll see the opening dialog box (refer back to Figure 7.5). Select the Next button; you are asked to select the databases for which you want to create the maintenance plan (see Figure 7.17). By default, this database is the selected database. In total, you have four types of databases from which to choose:

- **All databases** These are all the system and user-defined databases (as registered in the master database).
- **All system databases** These are the master, msdb, and model databases used by SQL Server 2000 to keep your user-defined databases operational.
- **All user databases** These are all databases except the system databases.
- **These databases** Under Databases, all available databases are listed. You can select the desired databases for the maintenance plan.

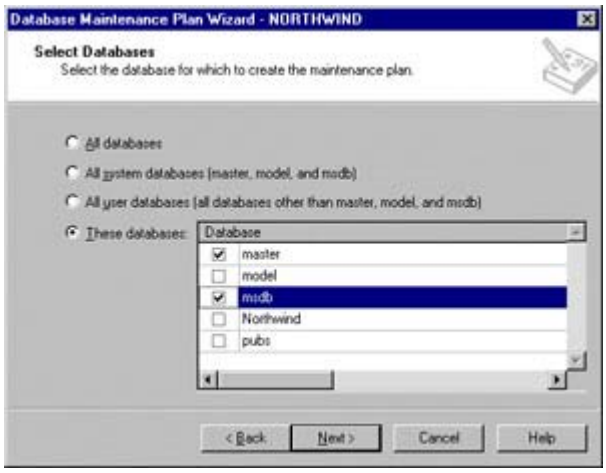


Figure 7.17: The Select Databases dialog box.

For our monthly maintenance plan job of the system databases:

1. Select the option "These databases" in the Select Databases dialog box.
2. Check the databases master and msdb.

3. Press the Next button.

Next you are prompted for a number of optimization options (see Figure 7.18):

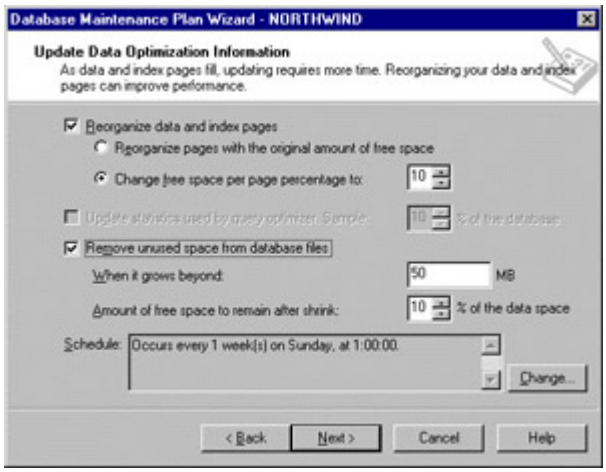


Figure 7.18: The Update Data Optimization Information dialog box.

1. **Reorganize data and index pages.** If you check this option, all indexes will be recreated after first being dropped. This reorganization of the indexes can increase performance on tables with many rows. You have to choose from two reorganize options:
    - **Reorganize pages with the original amount of free space** When the database was first created, the parameter FILLFACTOR was assigned with a value, indicating the percentage of free space that must be kept in order to have room for the indexes to grow. This initial value is used when you select this option.
    - **Change free space per page percentage to** Instead of using the initial FILLFACTOR value (in the case of Figure 7.18 , it is 10 percent), you can select your own value. If you expect the indexes (and database rows) to grow significantly over a short period of time, you can increase this percentage. If this is not the case, keep the default value, since too much reserved space slows down the retrieval of data from disk.
2. **Update statistics used by query optimizer.** This option is mutually exclusive with the first option, since the reorganization of indexes also updates the statistics used by the query optimizer. If you do not want to reorganize the indexes but want to accelerate the retrieval of rows during the execution of queries, you can check this option. Additionally, you can enter the percentage of database pages that are used to determine the statistics by entering a value for the “Sample % of the database” field.
3. **Remove unused space from the database files.** For performance reasons, you want the database to be as small as possible, but not too small, since that would decrease the performance if many rows are added. This option carries two parameters:
    - **When it grows beyond** Gives you the opportunity to determine the threshold beyond which the shrinking of the database should occur.

- **Amount of free space to remain after shrink** Determines the percentage of free data pages in the database you want to retain. You can enter the value in the "% of the data space" box.

4. If you have checked at least one of these options, you are allowed to enter the scheduling information. After selecting the Change button, you enter the Edit Recurring Job Schedule dialog box (refer back to Figure 7.12).

For our monthly maintenance plan job of the system databases:

1. Check "Reorganize data and index pages."
2. Select "Change free space per page percentage" and enter the value **15** .
3. Select "Remove unused space from database files"; the default values will do.
4. Press the Change button.
5. For Occurs, select Weekly.
6. For "Every week(s) on" enter the value **4** and check only Sunday.
7. For the Daily Frequency, select "Occurs once at" and enter the value **01:00** .
8. For the Duration, the default values will do.
9. Press the OK button.
10. Press the Next button.

Now we can choose from among the Data Integrity Check options (see Figure 7.19):

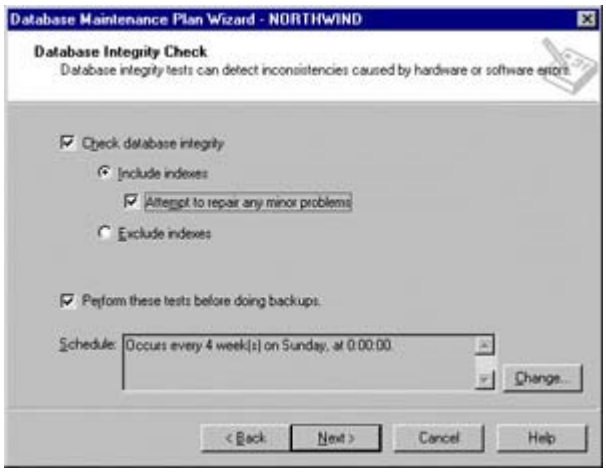


Figure 7.19: The Database Integrity Check dialog box.

- **Check database integrity** Checking this option means that you want an integrity test to be performed. This is done by running the DBCC CHECKDB Transact-SQL statement. You also have to decide if you want to include or exclude indexes. If you decide that the indexes should be included during the integrity check, you can check Attempt to repair any minor problems to let the detected minor errors be corrected. It is highly recommended to always check this option.
- **Perform these tests before doing backups** When this option is checked, the wizard takes care that the maintenance job first performs the database integrity control on the databases before the backup job is executed. If uncorrectable errors

are detected, the backup will not be performed. It is recommended that you check this option.

- **Check database integrity** If you select this option, you can select the schedule. In addition, you will enter the Edit Recurring Job Schedule dialog box (refer back to Figure 7.12) after pressing the Change button.

For our monthly maintenance plan job of the system databases:

1. Check “Check database integrity.”
2. Select “Include indexes.”
3. Check “Attempt to repair any minor problems.”
4. Check “Perform these tests before doing backups.”
5. Press the Change button.
6. For Occurs, select Weekly.
7. For Every week(s), on enter the value **4** and check only Sunday.
8. For the Daily Frequency, select “Occurs once at” and enter the value **00:00** .
9. For the Duration, the default values will do.
10. Press the OK button.
11. Press the Next button.

Notice that the default time of the schedule is an hour earlier than that of the optimization step. The wizard makes this suggestion, since it is the logical sequence.

The next two steps create the actual backup plan for the databases and the transaction logs. These two steps are nearly similar; for that reason, we look in detail at only the second one. The Transaction Log Backup Plan dialog box (see Figure 7.20) offers the following options:

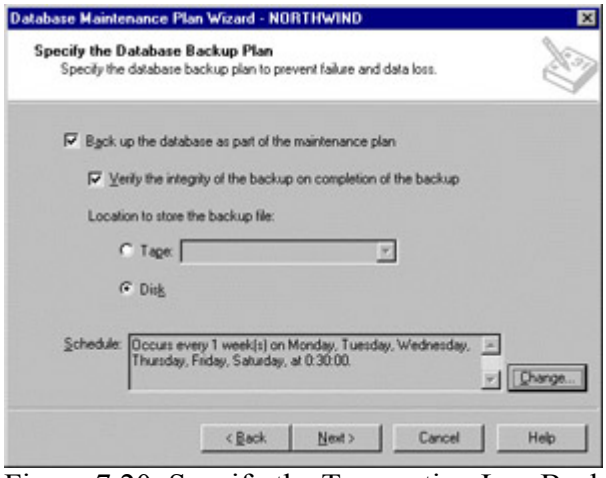


Figure 7.20: Specify the Transaction Log Backup Plan dialog box.

- **Back up the transaction log as part of the maintenance plan** Not checking this option means that the transaction log backup will not be part of this maintenance plan. However, since it is wise to do a daily backup of the transaction logs, you should check it.

- If you decide to make the transaction log backup part of this maintenance plan, you can:
    - Check “Verify the integrity of the backup on completion of the backup,” guaranteeing that if you have to rely on this backup, it is usable.
    - Select the location where the backup file will be stored: Tape or Disk.
    - Make a Schedule; via the Change button, you will enter the Edit Recurring Job Schedule dialog box (refer back to Figure 7.12).

For our monthly maintenance plan job of the system databases (for the transaction log backup):

1. Check “Back up the transaction log” as part of the maintenance plan.
2. Check “Verify the integrity of the backup on completion of the backup.”
3. Select Disk.
4. Press the Change button.
5. For Occurs, select Weekly.
6. For “Every week(s) on,” enter the value **1** and check all days except Sunday.
7. For the Daily Frequency, select “Occurs once at” and enter the value **00:30** .
8. For the Duration, the default values will do.
9. Press the OK button.
10. Press the Next button.

Assuming that the transaction log will be backed up on disk and you selected Disk as the appropriate location, you will be prompted with the Specify Transaction Log Backup Disk Directory dialog box (see Figure 7.21). Through this dialog box, you can choose these options:

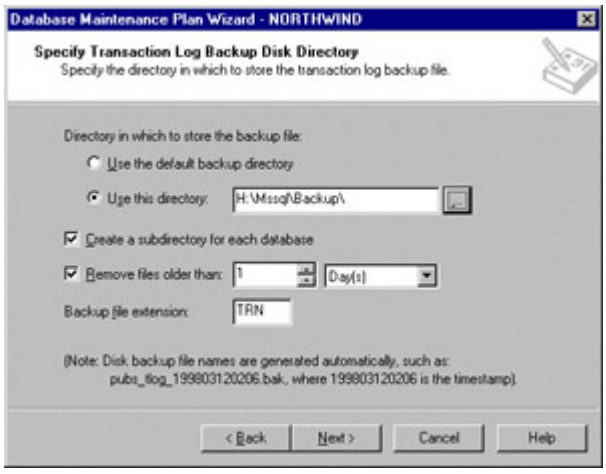


Figure 7.21: The Specify Transaction Log Backup Disk Directory dialog box.

- **Directory in which to store the backup file** This can be the default backup directory or another one you specify. The given default value is equal to the default backup directory.
- **Create a subdirectory for each database** With this option, you can keep the backup files of the databases that are part of the maintenance plan separate from

each other. Checking this option will create subdirectories in the selected backup directory. The name of the directory is equal to the database name.

- **Remove files older than** To keep control over the backup files written to the directory, you can let the maintenance plan remove older (out-of-date) backup files. The default value is 1 and week(s).
- **Backup file extension** The default for this option is this TRN for transaction log backup files and BAK for database backup files. The names of the backup files are generated automatically with a fixed structure. For the transaction log backup file, this structure <Database Name>_tlog_ <date><time>.TRN and the database backup filename structure is <Database Name>_<date><time>.BAK.

For our monthly maintenance plan job of the system databases for the transaction log backup:

1. Select "Use this directory" with the value H:\MSSQL\BACKUP
2. Check "Create a subdirectory for each database."
3. Check "Remove files older than" and enter the values **1** and **Day(s)** .
4. Use the default backup file extension.
5. Press the Next button.

The step following the creation of the database backup plan is report generation. Since this step speaks for it self, we won't discuss it in detail. As a last step you will be prompted for maintenance history (see Figure 7.22). The interesting point is that you always keep the "Write history to" option checked; otherwise, you won't be able to check for possible errors in the past. The "Limit rows in the table" option helps you prevent the history table from growing forever. The default value is 1,000, meaning that after 1000 rows, the space of the oldest rows is "reused." A cyclic history log is created.

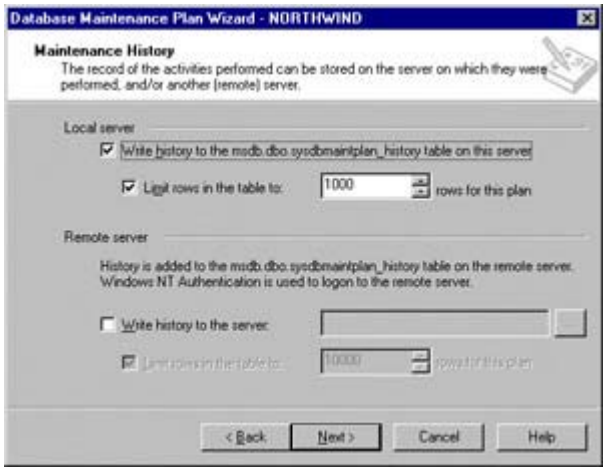


Figure 7.22: The Maintenance History dialog box.

The last dialog box is the Completing the Database Maintenance Plan Wizard, summarizing all selected and entered values. You only need the plan name to complete it. For our monthly maintenance plan job of the system databases (for the transaction log backup):

1. Check “Write report as a text file in directory” and use the proposed default value.
2. Check “Delete text report files older than” with the values 4 and Week(s).
3. Press the Next button.
4. Enter the plan name as **System databases (monthly)** .
5. Press the Finish button.

The maintenance plan will appear in the Enterprise Manager under <Your_Database_Server> | Management | Maintenance plan. If you select the plan and look through it, you will notice the tabs in the maintenance plan window resemble the steps the Database Maintenance Plan Wizard took. However, this maintenance plan is just a description. The execution of this maintenance plan can be found in the job list under <Your_DB_Server> | Management | SQL Server Agent | Jobs. The maintenance plan we created resulted in the creation of four jobs:

- Integrity Checks Job for DB Maintenance Plan “System databases (monthly)”
- Optimization Job for DB Maintenance Plan “System databases (monthly)”
- DB Backup Job for DB Maintenance Plan “System databases (monthly)”
- Transaction Log Backup Job for DB Maintenance Plan “System databases (monthly)”

If you take a closer look at the last job and look what the command for Step 1, it looks similar to this:

```
EXECUTE master.dbo.xp_sqlmaint
    '-PlanID 6F61E247-0102-4F7B-B3ED-76CF42CD977A
    -Rpt "G:\MSSQL\LOG\System databases (monthly).txt"
    -DelTxtRpt 4WEEKS
    -WriteHistory
    -VrfyBackup
    -BkUpOnlyIfClean
    -CkDB
    -BkUpMedia DISK
    -BkUpLog "H:\MSSQL\BACKUP"
    -DelBkUps 1DAYS
    -CrBkSubDir
-BkExt "TRN"'
```

The job executes the extended stored procedure xp_slqmaint, which is part of the master database. This procedure uses the xpstar.dll. The utility sqlmaint.exe uses this same library. The command has a great number of parameters. The ones used here are explained in Table 7.5 .

Table 7.5: The sysmaint Extended Stored Procedure and Utility Parameters

Parameter	Purpose
-PlanID	This parameter specifies the identifier of the maintenance plan. Sqlmaint uses this parameter to retrieve the databases to which this command must be applied. The PlanID must have a row in the table sysdbmaintplans in the msdb database.

Table 7.5: The sysmaint Extended Stored Procedure and Utility Parameters

Parameter	Purpose
-Rpt	The file to which the report of this execution of this command will be written. The filename must be accompanied by the full path for the local server or the UNC name for remote (or local) servers. To retain the other reports, sqlmaint inserts data and time in the filename. For example: \\DBServer\Mssql\report\System databases (monthly)200010271030.rpt
-DelTxtRpt	This parameter states that report older than the specified time period (in this case, four weeks) will be removed. The same full path or UNC name will be used as specified in –Rpt. The files matching the filename in –Rpt (time and date are replaced with the wildcard) and older than the specified time period are removed. Example: \\DBServer\Mssql\report\System databases (monthly)*.rpt
-WriteHistory	The parameter specifies that a history record is written in the table sysdbmaintplan_history in the msdb database.
-VrfyBackup	This parameter forces sqlmaint to run the RESTORE VERIFYONLY command.
-BkUpOnlyIfClean	With this parameter, a backup is made if the database has no problems. A parameter checking the database (in this case,–CkDB) must be specified; otherwise, a backup will take place anyway.
-CkDB	Sqlmaint is forced to execute the command DBCC CHECKDB. This parameter must be specified earlier than –BkUPMedia or –BkUpLog if the parameter –BkUpOnlyIfClean is specified. If this is not the case and –CkDB reports an error, the backup will already have been made and –BkUpOnlyIfClean will have no effect.
-BkUpMedia	This specifies the type of media used to back up: Disk or Tape.
-BkUpLog	This parameter is used if a transaction log backup has to be made, followed by the path of the backup device. For database backups, –BkUpDB is used. The name of the backup file is: <Database name>_tlog_<date><time>.< Extension> The default extension is .BAK, unless –BkExt is specified. Example: Master_tlog_200010271030.TRN

Table 7.5: The sysmaint Extended Stored Procedure and Utility Parameters

Parameter	**Purpose**
-DelBkUps	If this parameter is used, backup files older than the specified time period (in this case, one day) are deleted. The path specified in –BkUpLog/-BkUpDB is used to determine the existing backup files. Notice that this parameter is valid only if the backup device, specified with –BkUpMedia, is Disk.
-CrBkSubDir	With this parameter, a subdirectiory is created for every database. Again, this is valid only if the backup device is Disk.
-BkExt	This specifies the extension of the backup filename.

Restoring SQL Server Databases

The restore of a database in SQL Server 2000 can be performed in a very effective way using the capabilities incorporated in the SQL Server Enterprise Manager. All backup history is recorded in the msdb database. You have full use of this information. In fact, SQL Server will even suggest which complete, differential, and transaction log backup you should restore to rebuild the database to be as up to date as possible. You can even automate restores with the use of the Transact-SQL statement and a scheduled job.

Note In contrast to the backup, a restore can be performed only if no users are connected to the database.

Restoring a Database Backup

The easiest way of restoring the Northwind database is to use the Enterprise Manager. Select the Northwind database in the tree under <Your_Database_Server> | Databases. Then select Tools | Restore Database. This will open up the Restore Database dialog box (see Figure 7.23).

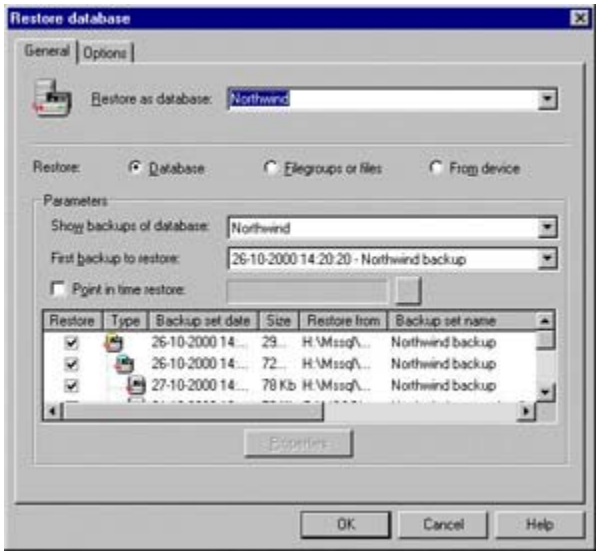


Figure 7.23: The Restore Database dialog box.

The Restore Database dialog box has two tabs: General and Options. The layout of the General tab is related to the type of restore (Database, Filegroups or Files, From Device) you will select. Let's take a look what other kind of information is available on the General tab:

- **Restore as database** You can select the database to which the restore must take place. In most circumstances, the backup files are made from this same database. However, you can create a new and empty database and then restore the backup files of an existing database, effectively creating a database copy. The selection

field shows all the registered databases from which you can choose. The default value is the currently selected database in the Enterprise Manager.

- **Restore** This option lets you choose from three types of restore:
  - **Database** Uses the backup history of the selected database as recorded in the msdb database. If you select this option, the list of available complete, differential, and transaction log backup files are shown. Database is the default restore type.
  - **Filegroups or files** As mentioned earlier in this chapter, you can restore specific database files or filegroups. If you select this option but never added files or filegroups, a list with the backups of the primary database files is shown (see Figure 7.24).

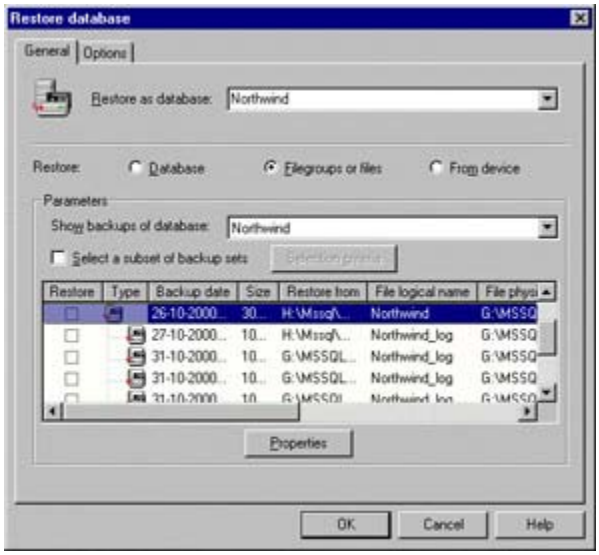


Figure 7.24: Restore filegroups or files.

  - **From device** In this case, you do not rely on the backup history in msdb but read the contents of a device (normally a tape device) and base the restore on the backup files on tape. You can also use this option to reconstruct the backup history, in case the msdb got lost (see Figure 7.25).

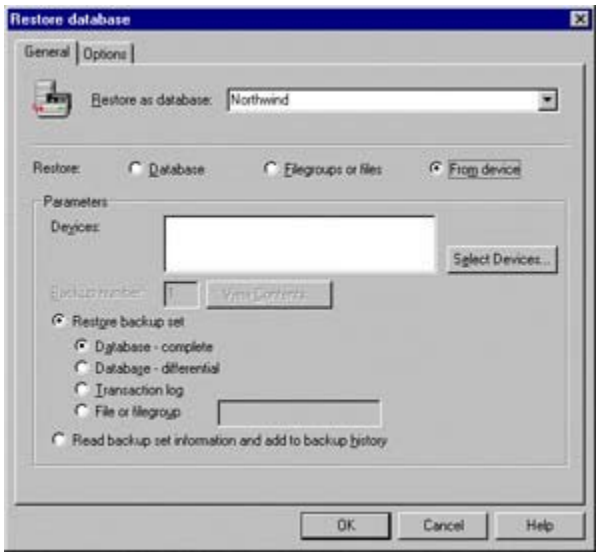


Figure 7.25: Restore from device.

Depending on the restore type, the set of parameters changes (as Figures 7.23 through 7.25 show). To be able to use the RESTORE command, it is important to have a good grip on the meaning of these parameters.

For the database restore type, the parameters are:

- **Show backups of database** In most cases, this will be the same data-base as specified in the "Restore as database" field. However, if you want to create a copy of an existing database based on its backup files (you can even restore an existing database to an previous version), this can be the name of that particular database.
- **First backup to restore** By default, the value is set to the last complete backup of the selected database. Based on the provided list of complete backups (which is based on the available backup history), you can select the one from which you want to restore.
- **Point in time restore** If transaction log backup files are available, you can select a specific date and time to which you want to restore completed transactions.
- **Restore list** This field lists all available backup files based on the selected complete backup (in the field "First backup to restore"). The complete backup is on top of the list, followed by all the differential and transaction log backups. By default, the complete backup, last differential backup, and all transaction log backup files made after the last differential backup are checked as candidates for restore. This selection is based on the presumption that you want to restore the database to the latest possible state. In Figure 7.23 , the complete backups of one differential backup and one transaction log backup are shown, whereby the complete and last differentials are checked by default. Of course, you can uncheck the last differential and check the prior one if you suspect that the last differential contains referential integrity errors.
- **Properties** After you select a row from the restore list, the Properties button becomes enabled, whereby you can recall the properties of that backup file.

Notes Back up the transaction log before restoring. If you want to restore a database to the latest possible state, you should first make a backup of the transaction log. If you forget to do that, you will lose all transactions that took place from the last backup to the point at which you decided to restore the database.

You can do this by first selecting the appropriate database, and then selecting Tools | Backup Database. The Backup Database dialog box will appear. You should do the following:

- Verify that the database name is correct and change it if needed.
- Enter an appropriate name for this backup.
- Under Backup, select "Transaction log."
- If the Destination field contains an existing backup file, remove it by pressing the Remove button. Then press the Add button, select the backup directory, and enter a new filename, preferably with the standard format <database name>_tlog_<date><time>.TRN.
- Under Overwrite, select "Overwrite existing media."
- Press the OK button.

Now the transaction log backup will be made and the transaction log will be truncated.

For the filegroups or files restore type, the parameters are:

- **Show backups of database** In most cases, this will be the same database as specified in the "Restore as database" field.
- **Select a subset of backup sets** With this option, you can limit the number of complete backups appearing in the list. This option is useful if you have history that goes back a long time. By checking this option, the Selection criteria button is enabled. The selection criteria are based on the place of the backup files, backup files not older than a certain date and time, or specific filegroups or files.
- **Restore list** This lists all complete backups that adhere to the selection criteria. By default, this includes complete backups of all filegroups and all files that are recorded in the backup history.

For the filegroups or files restore type, the parameters are:

- **Devices** Select the device(s) from which you want to restore backup files. If you have more devices from which you need to restore backup files, you can select these all so that the backup set on these media can be scanned.
- **Backup number** This is the backup set number on the device—for example, the cartridge in the tape unit you want to restore. If you do not know the specific number, you can use the View Contents option. The complete media is scanned for backup sets, which are listed, enabling you to select the appropriate one. Be aware that this can take some time, since tape units are relatively slow devices.

- **Restore backup set** You must select the type of backup file in the backup set you want to restore. This type can be Complete, Differential, Transaction log, or File(group). With File(group), you also need to enter the name of the specific file or group.
- **Read backup set information and add to backup history** Instead of restoring a backup set, you can add information over the backup files on the backup set (for example, the tape cartridge) to the backup history. This option is useful if you receive backup tapes made on a different SQL database server—or worse, you have to reconstruct the backup history because you cannot restore the corrupted/lost msdb database.

Restoring System Databases

Suppose you need to copy the system databases to another database server, overwriting the system databases on that server. You can do this by backing up the system databases on the current database server and do a restore from this backup file on the other server. This can be achieved using the Restore Database option in the Enterprise Manager, as described in previously. You need to do this manually every month. Since you have a busy schedule yourself, it is a task that tends to be forgotten. The other option is to create a SQL Agent job (using the Create Job Wizard) and construct instructions using the T-SQL command RESTORE.

The following RESTORE T-SQL command restores the monthly master database backup, which was placed on disk to speed up the restore. The arguments are explained in Table 7.6 :

```
RESTORE DATABASE [master]
    FROM DISK = 'H:\MSSQL\BACKUP\Master Monthly (complete).BAK'
    WITH
           FILE
=
1

,


           MEDIANAME = 'Master_monthly',
           RECOVERY,
           NOUNLOAD,
           REPLACE,
STATS
=
 10
```

This RESTORE command is the same for complete and differential backup files. The restore process restores the database page by page, independent of whether or not the backup file contains all pages or a limited set of pages.

If you need to restore the last transaction log of the msdb database using the RESTORE command, it will look like this:

```
RESTORE LOG [msdb]
   FROM TAPE = \\.\TAPE01
   WITH
          FILE
=
 1,
          MEDIANAME
=
Msdb_daily',
          RECOVERY,
UNLOAD
```

The RESTORE command can also be used to restore a specific database file. Although the master database has only one database file (master.mdf), it is possible to restore this file with the following command:

```
RESTORE DATABASE [master]
   FILE = 'master.dbf'
   FROM DISK = 'H:\MSSQL\BACKUP\Master Monthly (complete).BAK'
   WITH
           FILE
=
1

,


           MEDIANAME = 'Master_monthly',
           RECOVERY,
           NOUNLOAD,
           REPLACE,
STATS
=
 10
```

Table 7.6: RESTORE Command Arguments

Argument	**Explanation**
RESTORE DATABASE	If the RESTORE command is followed by DATABASE, the command needs to restore a database file (complete or differential).
RESTORE LOG	If the RESTORE command is followed by LOG, the specified transaction log will be restored. If the transaction log contains incomplete transactions, these will not be committed, unless the option NORECOVERY is specified.
FILE	This argument specifies the name of the database file you want to restore.
FROM DISK	This argument defines that you want to restore from a backup

Table 7.6: RESTORE Command Arguments

Argument	Explanation
	file that is stored on disk. The complete file path is provided. Instead of DISK, you can use TAPE for a tape device or PIPE for a named pipe.
(WITH) FILE	This argument specifies the backup set that has to be restored. Since a backup medium can hold more backup sets, you have to define which you need to restore.
UNLOAD	This argument triggers the tape to be ejected from the tape unit after the restore. NOUNLOAD stays active for all subsequent BACKUP commands until a BACKUP command uses the UNLOAD argument.
RECOVERY	Instructs the restore operation to roll back any uncommitted transactions. After the recovery process, the database is ready for use.
REPLACE	With this argument, you can force a database file to be overwritten with the one from the backup device. If you do not use the REPLACE argument in the RESTORE DATABASE command, a check will be performed to ensure that no invalid or unwanted restore will take place. This check fails if: • The database in the RESTORE command is already registered on the database server and the database name does not match the database name in the backup set. • The number of database files of the database on the server differs from the number of database files in the backup set. Size is always ignored, since databases can grow or shrink. The REPLACE command is useful in situations in which you want to copy or move a database and you need to overwrite or replace the existing database (files) anyway.
STATS	With this argument, you are notified of the progress of the restore. The value specified is the percentage of this progress reported. The default value is 10, meaning that after every 10 percent, the progress is updated.
LOADHISTORY	With this argument, all information on the read backup set is loaded in the backup history tables of the msdb database. Only backup sets that are related to backups made through SQL Server can be recorded in the history tables.

Table 7.6: RESTORE Command Arguments	
Argument	**Explanation**
MEDIANAME	This argument carries the name of the media. If specified, it must match the media name used on the media; otherwise, the restore will fail.

The RESTORE command has four subcommands that can be used to inspect backup files. These commands are particularly useful in situation in which you do not have a clear picture of the tape or backup file contents. For an explanation of the arguments, refer back to Table 7.6 .

- **RESTORE FILELISTONLY** This command scans the backup device and returns a list of the database and transaction log files that are contained in the backup set. Example:

```
RESTORE FILELISTONLY
             FROM TAPE = \\.\TAPE1
             WITH
                   FILE = 1,
      UNLOAD
```

- **RESTORE HEADERONLY** This command scans the complete backup device and returns a list of the headers of every backup set that exists on the backup device. Take notice of the fact that the complete tape is scanned. It can take a while before a large tape is fully read. Example:

```
RESTORE HEADERONLY
              FROM TAPE = \\.\TAPE1
              WITH
     UNLOAD
```

- **RESTORE LABELONLY** This command scans only the beginning (header) of the backup device and returns a single line of label information. Since only the beginning of the device is read, the command returns quickly with the result. Example:

```
RESTORE LABELONLY
             FROM TAPE = \\.\TAPE1
             WITH
     UNLOAD
```

- **RESTORE VERIFYONLY** With this command, two functions can be performed. The first one is to check for validity and correctness of a backup set, resulting in a message stating that the backup set is valid or invalid. The second function is to load the information on the verified backup set into the backup history in the msdb database. In this way, you can construct a history of backups that were never part of the history recorded in the msdb database. Example:

```
RESTORE VERIFYONLY
               FROM TAPE = \\.\TAPE1
               WITH
                     FILE = 1,
                     UNLOAD,
   LOADHISTORY
```

Database Options and Settings

From within the Enterprise Manager, a few database settings and SQL Server database settings are directly related to the backup and restore process. To be able to back up the transaction log, you should prevent it from being truncated automatically. This is done by taking care that the "Truncate log on checkpoint" setting is set to false. You can change the value of this option this way:

1. Select the appropriate database in the Enterprise Manager.
2. Right-click the Properties option. This will bring the Properties dialog box on screen (see Figure 7.26).

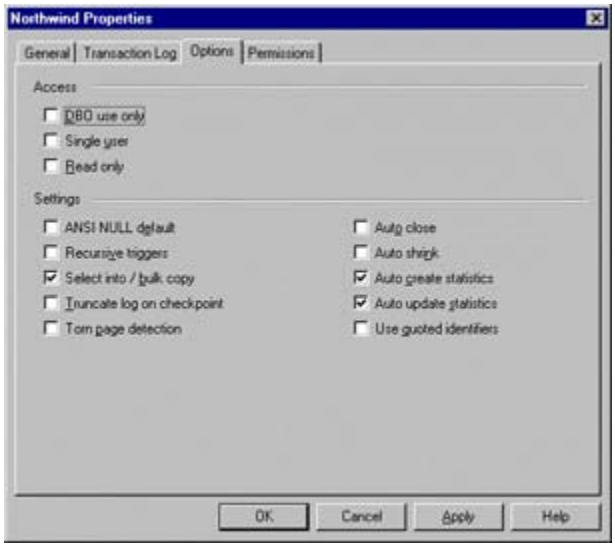


Figure 7.26: The Northwind Properties Options tab.

3. Select the tab Options.
4. Verify whether or not the option "Truncate log on checkpoint" is checked in Settings.
5. If the option is checked, uncheck it.

This options needs to be set or unset for every database that is accessible from the database server.

Within the properties of the SQL Server, you can set a few options that have an effect on all databases available on the database server. The first setting has to do with the amount of time the SQL Server waits for a tape to become available from the tape device, after a

backup or restore operation is started. The time-out period (see Figure 7.27) can have three values:

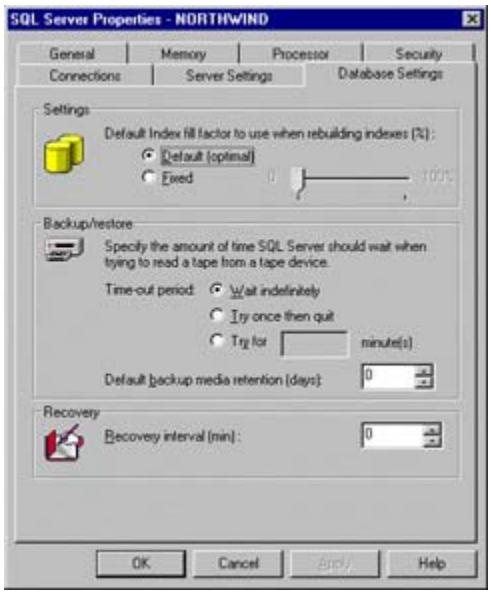


Figure 7.27: SQL Server Properties Database Settings tab.

- **Wait indefinitely** This is the default value and tells the backup/restore process to keep checking to see if a tape is inserted. This option locks up system resources and other operations from accessing the tape device. This option is very convenient for an attended backup or restore.
- **Try once then quit** The backup/restore process will check for a tape to be ready for use. If no tape is ready, it aborts the operation and will lock the failure. You can use this option for unattended backups, where you expect a tape to be present.
- **Try for … minute(s)** With this option, you can satisfy your needs for both attended and unattended backup/restore operations. It can provide you enough time to insert or change a tape, but it will not lock the process indefinitely during an unattended operation.

A second setting is the "Default backup media retention (days)." This is the default value you can use when creating a database backup job (refer back to Figure 7.10), with the field "Backup set will expire after." The value is zero by default. By increasing the field (equal to RETAINDAYS), you can prevent any backup set from being overwritten right away. Even if you manually backed up a database, this value will be used if it is greater than zero.

The "Recovery interval (min)" setting affects the backup of the transaction log. This setting specifies the number of minutes the SQL Server is granted for the recovery of a database. The interval is used on a per-database basis. This value tries to control the size of the active portion of the log. The larger the active portion, the greater the loss of data can be after a failure, since the amount of data that can be backed up is smaller. The default value is 0 minutes, which means that SQL Server will configure automatically, depending on the amount of data changed.

Testing Your Backup and Recovery Strategy

Your restore is only as good as your last backup. Making backups is a daily chore, eventually prone to undetected technical and/or human errors. You have every reason to test both your backup and restore strategies. Attending to the backup logs and changing tapes is not the most challenging job, so these activities are regularly handed over to another person, thereby degrading the continuity and reliability of the way the job is handled. We forget that the tapes can hold your company's continued existence in case of a disaster.

The only way of preventing the complete loss of your vital data is periodic testing. One possible scheme involves planning a regular backup and restore test at least four times a year. You can prepare these tests ahead of time and take care that they are done thoroughly. However, performing a real-life restore test can leave shortcomings in the restore procedure. This kind of test should always be done under a time regime and must not be planned but announced at random, similar to a building fire evacuation drill. The head of the computer or IT department is responsible for announcing the problem. He or she should also be responsible for creating the failure scenario. A good example for a scenario is that the database server has failed and cannot be rebooted. The only prerequisite is that there is a database server configuration at hand. This could be hardware normally used for testing purposes, or a configuration can be rented for a few days.

You should build a test plan that exactly states what is tested, how it is tested, in which sequence, and how the test result is reported. The administrators must take turns executing the test plan. A healthy working climate must exist, one in which peer reviews are acceptable. It is better to learn from the test and collectively make improvements to the problem areas than to point fingers or state that someone in particular has done something wrong.

The test plan should address the following questions:

- Are all necessary backup jobs running? Are they doing the actions they are suppose to do? In other words, are the backup jobs implemented as stated in the backup strategy?
- Do the SQL Server, application, and/or system logs contain errors or warnings related to the backups? Is action already taken based on these errors or warnings?
- Are the hourly transaction log and six-hour differential backups placed on the disk storage of another server?
- Are all tapes in the data vault except for the one on which the next backup must be recorded?
- Has the logbook been kept up to date?
- Are the heads of the tape units regularly cleaned?
- Choose a few tapes at random. Check the content of the tapes, and try to restore one or two files on the tapes. This is a technical check of the proper working of the backup.

- Perform a complete restore of the database based on the complete backup from a week ago, the incremental backup the following day, and the transaction log backups of the following 24 hours (these should reside on one of the tapes of the regular system backup or the server holding the transaction log backup files). Time every restore that is made so that you get a better estimate of the duration of restores.
- Make a manual differential backup of the database to a new tape cartridge and record the duration.

Summary

The availability of your database solution is guaranteed only if an adequate back-up and recovery strategy is in place. The best approach is to plan and implement a backup and recovery strategy even before you deploy your database. You need to determine the risks you will be confronted with if you are not able to restore the database. In most cases, these risks are too grave to ignore. This means you need to determine the requirements to put in place a reliable backup and recovery plan. This plan must be controlled through the proper procedures in handling tape rotation, especially if you use an offsite secure storage facility. Additionally, you need to come up with a test plan that enables you to test the reliability of the backups, restores, and the procedures. It is an absolute necessity to continuously affirm that you are able to restore the database to the minimal required state and in the shortest possible time.

SQL Server 2000 incorporates a number of facilities that enable you to quickly implement backup schemes and a high level of support in restoring databases. The Create Database Backup Wizard helps you create a scheduled backup job for a database. The Database Maintenance Plan Wizard goes a step further, also allowing you to create scheduled jobs for optimization and consistency checks of a number of databases. The maintenance plan jobs use the sysmaint utility, whereas the backup jobs use the Transact-SQL commands BACKUP or RESTORE. These commands can be used to create elaborate backup and restore scripts. Using the Enterprise Manager, you can restore a database. You can then easily use the backup history contained in the msdb database, which lets you select the appropriate complete and differential backups. Moreover, by taking care that the transaction log backups are available, you can restore a database to the state of a specific time.

FAQs

Q. Do I need to make backups of my database every day?

A. Yes. In nearly every case, a database holds your organization's valuable information that you cannot afford to lose. Needing to restore the database with as little loss of data as possible forces you to make a backup at least once a day. In fact, making more backups a day is even better. Doing so is by any means cheaper than recreating data lost in a failure. Do not rely too much on the infallibility of your hardware.

Q. How can I preserve as much data as possible after a failure of the database server?

A. A number of possibilities enable you to preserve as much data as possible. First, you need a database server configuration that can sustain a single failure. Second, distribute the system databases. Place user-defined data-bases, transaction logs, and backup files on physically independent storage devices. Third, make a tightly fitting backup scheme—for example, weekly complete backup, a daily differential backup, and hourly transaction log backups. If you can preserve all backup files after a failure, you will be able to restore the database with at most one hour of data loss. However, if the database server is still accessible, it is very possible that the current trans-action log can be safely backed up, preserving even more data.

Q. What is the difference between the Create Database Backup Wizard and the Database Maintenance Plan Wizard?

A. Both wizards are equally good at the job they are designed for. The Create Database Backup Wizard helps you create a single backup job for a single database. The Database Maintenance Plan Wizard helps you create more jobs for a number of selected databases. The jobs that are created by the Data-base Maintenance Plan Wizard are in addition to the backup of databases and transaction logs as well as the optimization and error checking of the databases. A second important difference is that the jobs created by the Create Database Backup Wizard use the Transact-SQL command BACKUP, whereas the jobs of the Database Maintenance Plan Wizard use the extended stored procedure xp_sqlmaint. If you would use all functionalities provided by the Database Maintenance Plan Wizard, it is recommended that you use this wizard for the differential backups, and the Create Data-base Backup Wizard can be used to make the complete backups. However, remember to schedule the jobs in the right order: (1) consistency check; (2) optimization; (3) differential backup databases; (4) backup transaction logs.

Q. What is the best approach to restoring a database?

A. If the restore is necessary due to a system failure or a corrupted database, make sure that the cause of this failure corruption is well understood. If you simply went ahead with the restore, it is very likely that you would soon run into the same problem again. So first, assure yourself that the technical cause of the failure is repaired. Before bringing up the SQL Server, prevent users from being able to access the database. At this point, it important to know if the master and msdb databases are still in working order; otherwise, these must be restored first. Then check to see if you can still back up the transaction log of your database using the backup tool of the Enterprise Manager. After that, use the restore tool of the Enterprise Server to restore the database. Based on the backup history in the msdb database, it makes a

suggestion as to the complete, differential, and transaction log backups you should restore. Go ahead with the suggested restore. After that, check the database thoroughly before bringing it online again.

Q. I use replicated databases. How should I back up and restore them?

A. The best backup strategy for replicated databases is the same as the one you use for autonomous databases. In the case of transactional and snapshot replicated databases, the restore process is very straightforward. If the publisher or distributor failed, they can be restored the same way as an autonomous database, followed by enabling the synchronization between them. Subscriber databases can be deleted and removed from the publisher or distributor and recreated, triggering a full replication. If you use merge replication, you have a situation in which a subscriber is in fact a publisher. Therefore, synchronization must occur both ways. After restoring the database with a minimal loss of data, synchronize it with the other publisher first, and then resynchronized the publisher with the restored database.

Q. How can I change my backup jobs?

A. If you are not an experienced SQL Server administrator, the best way is to use the wizards to create a new backup job first and then remove the old one. Do not do it the other way around. Remember that if the job concerns a maintenance plan, you must also remove the registered maintenance plan. More experienced administrators can directly modify the SQL Agent jobs that are created with the Create Database Backup Wizard. If you used the Database Maintenance Plan Wizard, you should make modifications to the maintenance plan only. The Enterprise Manager takes care of modifying the SQL Agent jobs.

Q. Can I restore my database automatically?

A. Technically, yes. However, practically, it will not work most of the time, since you might need more tapes to be able to make a full restore. Additionally, it is important that decisions in the restore process are double-checked, which is not possible if the process is fully automated. However, if you want to restore a database automatically, you need full knowledge of the T-SQL RESTORE command. You can use the Create Job Wizard to create a SQL Agent job. During the execution of the wizard, you can create a T-SQL step. By inserting the complete RESTORE command in the Command field and creating the desired schedule plan, you can restore a database automatically. Do not forget that a restore can be executed only if no one is using the database. Again, refrain from restoring a database automatically; doing so could result in more work than anticipated.

Chapter 8: Microsoft English Query and Full-Text Search

Introduction

Locating information in SQL Server is as important as storing it. SQL Server 2000 goes beyond its standard support for Structured Query Language (SQL) with full-text search and English query capabilities. Each of these two components has a distinct role in adding value to the data stored in SQL Server. This chapter reviews their configuration and use in detail; you will learn how to configure each of these features for use in your applications.

Microsoft English Query is an optional component included with SQL Server and can be installed to take advantage of free-form English search capabilities. Users can be allowed to ask questions using common English expressions and receive their results from the database server. This capability allows developers to extend their applications to include ad hoc search capabilities in solutions such as decision support systems. Instead of understanding and issuing complex SQL statements, users can ask questions such as "How many widgets did we sell last month in the Northeast?" and receive valid results from SQL Server. With usability in mind, Microsoft English Query has been enhanced to decrease development time, with Visual Studio integration and new wizards to create basic English Query projects.

With the addition of the text data type in SQL Server 7.0, support for searching and locating information in text fields has become the task of the *Full-Text Search* component in SQL Server. Full-text searching allows users to locate specific and pattern-matched content in any text field using T-SQL statements. Using full-text searching allows you to extend your applications to take advantage of complete document text storage and retrieval. Applications such as knowledge base solutions, user support, and e-commerce catalogs can benefit from full-text search capabilities.

Overview of English Query

Adam Smith, in his landmark work *The Wealth of Nations* , went into great detail arguing that each person should do what he does best. Following his logic, marketing specialists should be engaged in marketing, accountants should deal with accounts payable and receivable, and managers should manage. All these groups need information to accomplish their tasks; much of this information can be contained in SQL Server, but none of these groups should be writing SQL or Multidimensional Expressions (MDX) statements. This is the idea behind English Query: Let people do what they do best by allowing them access to the information in database servers using their natural spoken language.

Letting people do what they do best—that sounds great, so why hasn't it been done before? Well, it was! English Query was first introduced with SQL Server 6.5 Enterprise Edition in 1997. However, with many software products, it typically takes two or three versions before the capabilities mature and people are comfortable working with them. English Query 2000, the third version of English Query, is a product of great power with which it is effortless to develop.

It is easy to see why it took time to develop an English query processing engine and usable development tools. After all, writing applications that are able to translate natural language into database queries sounds like a daunting task. Perhaps it would be if you had to develop all the pieces yourself; most of us don't have detailed enough knowledge of the English language to create the framework to translate the spoken word to SQL. Of course, with enough resources and time, we could do it. But shouldn't we be doing what we do best instead? So that we can do what we do best (writing COM-based databases applications), Microsoft has built as series of COM objects and development tools, collectively called Microsoft English Query. With these COM object and tools, writing applications that translate natural-language queries into database queries has moved from the realm of science fiction to practical reality.

As illustrated in Figure 8.1 , the core component of Microsoft English Query is the English Query run-time engine. The run-time engine is a programmable set of COM objects that are able to load a semantic model and use it to transform natural-language questions into standard SQL data-base queries. To take advantage of the work that Microsoft has done with the English Query engine, you must create a semantic model representing your database entities in an English Query Domain (*.eqd) file. You can create these semantic models using the development tools that are included with Microsoft English Query.

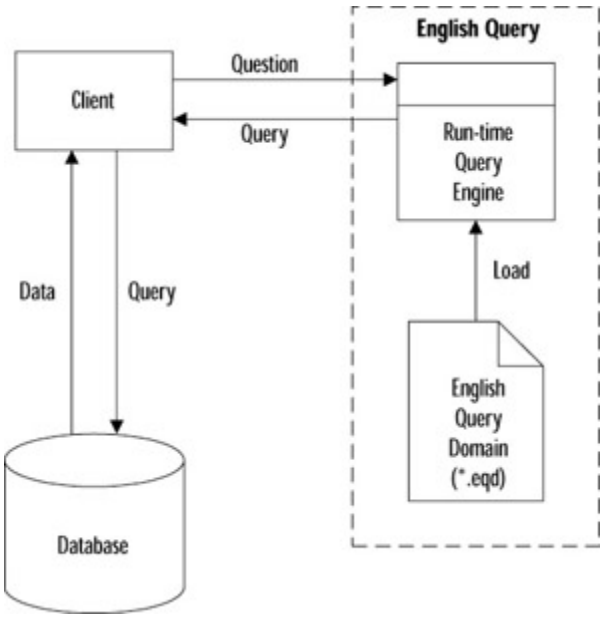


Figure 8.1: English Query run-time architecture.

After you have developed your semantic model, you can integrate Microsoft English Query into your COM-compatible applications by including the run-time engine and using it to convert natural-language questions in to SQL statements based on your semantic model. Your application will use the English Query engine to translate the user's questions into valid SQL statements that your application can issue to your SQL Server database to retrieve the desired results.

What's New in English Query?

Much of the developer feedback that Microsoft received on the previous versions of English Query surrounded the complexities of building and deploying English Query solutions. With the power and reliability of English Query keeping it afloat, Microsoft focused on delivering a more consistent and developer-friendly environment for building and integrating English Query solutions. Some of the more significant new features in English Query 2000 are:

- Microsoft Visual Studio 6.0 Integration
- OLAP Project Wizard
- SQL Project Wizard
- Graphical Authoring
- New Regression Features
- Enhanced Authoring by Example
- Two-Click Deployment
- Graphical Question Builder

It seems like a reasonable hypothesis that many of the people who will author Microsoft English Query applications will be familiar with Microsoft Visual Studio. Microsoft decided to leverage the popularity and familiarity of Visual Studio by integrating English

Query development into the Visual Studio development environment. Leveraging the graphical development concepts behind Visual Studio, English Query includes drag-and-drop graphical authoring that allows easy manipulation and visualization of entities and relationships. This approach to consistency and the large community of Visual Studio developers will make English Query development an easier transition and will undoubtedly increase its popularity in SQL projects.

No new release of a Microsoft product would be complete without the addition of task wizards, and English Query follows this trend. There are two new and very important wizards: the *SQL Project Wizard* and the *OLAP Project Wizard.* The SQL and OLAP Project Wizards interrogate the data-base schema for information about tables and cubes and use that information to generate the entities and relationships into a basic English Query model. Microsoft estimates that the SQL Project Wizard will be able to model 70 percent of the relationships and that the OLAP Project Wizard will be able to model over 90 percent of the relationships in well-structured, normalized databases. In addition to these features, development has been made easier by improving the Authoring by Example feature to provide multiple suggestions.

In addition to making development of semantic models easier, the Graphical Question Builder and two-click deployment were added to simplify the development of client interfaces. The *Graphical Question Builder* is a querying interface for the end user that helps make building questions easier and more intuitive. It is a graphical tool that allows users to inspect all available relationships between the entities in the application's English Query model. *Two-click deployment* was added to provide deployment of an English Query project to the Web with little manual development or configuration. If you have Visual InterDev installed, two-click deployment will build a solution with all the required application service provider (ASP) code to enable your Web site to query the database with natural language.

Finally, Microsoft added one last feature to improve the development usability of Microsoft English Query: enhanced testing features. The regression features have been expanded to allow output to be saved in XML files, allowing them to be accessed and even generated by other tools. These new testing enhancements will help promote stability in the production of English Query applications, because you can use these test files to ensure that future development does not affect existing functionality.

More Powerful Applications

A great deal of effort went into making English Query easier to work with, but at the same time there were many new features added to increase its functionality:

- Semantic Modeling Format
- Authoring Object Model
- Analysis Services Integration
- Full-Text Query Support
- Oracle Database Support

XML is the glue that that binds the .NET platform together; you should expect to see it creep into every tool and product Microsoft releases, including this version of English Query. One of the areas in which XML is used in English Query is through the Semantic Modeling Format (SMF). SMF is an alternative format to describe the entire model that you can build in the English Query development tool. Because SMF is stored as XML, your models are available to all the tools that can manipulate and parse XML. One of these tools is the Authoring Object Model included in English Query 2000. The Authoring Object Model is a programmable COM object that can work with SMF, offering a different authoring environment than Visual Studio. With this programmable environment, you can create your own wizards, daemons, and tools to develop and maintain English Query applications.

Analysis Services Integration, Full-Text Searching, and native support for databases other than SQL Server were also added to increase the capabilities of English Query. For power users, Analysis Services Integration allows English Query to access dimension tables as multidimensional cubes through MDX. Full-text querying support allows English Query to generate SQL that takes advantage of predicates such as FREETEXT and CONTAINS. These predicates offer much better performance than using LIKE searches, as well as much more flexibility. In the past, the only way that you could get English Query to work with other platforms was through heterogeneous linking. You can continue to use heterogeneous linking, but you can also interface with other databases such as Oracle (version 7.0 and greater) and Microsoft Access natively, without using SQL Server heterogeneous linking.

Installing English Query

English Query is not a part of the SQL Server engine components; it is a development package that can be used to include natural-language query capabilities in your applications. As such, English Query is not automatically installed when SQL Server is installed. You must use the SQL Server Setup program and explicitly install English Query (see the Install Components screen in Figure 8.2). Once you verify that you meet the installation requirements (listed in the following section), you can install English Query using the following steps:

Installing English Query

1. If the machine on which you are installing English Query is running OLAP Services version 7.0, OLAP Manager, or an Internet Information Server (IIS) project that uses the English Query application, you must stop them during the installation process.
2. Double-click the SQL Server Setup program icon from the installation CD-ROM.
3. On the first screen, choose SQL Server 2000 Components to display the Setup screen shown in Figure 8.2 .



Figure 8.2: The Install Components screen in the SQL Server 2000 Setup application.

4. Choose Install English Query. Data Access Components will automatically be installed or updated.
5. The next dialog box will display the End-User Agreement. Review the agreement and click I Agree.
6. The next dialog box will ask you to choose between the Complete and Run-time Only installation options, as shown in Figure 8.3 . If you are going to create English Query projects, choose the Complete option. If you will be running

English Query applications but will not need the development tools , choose the Run-time Only option .

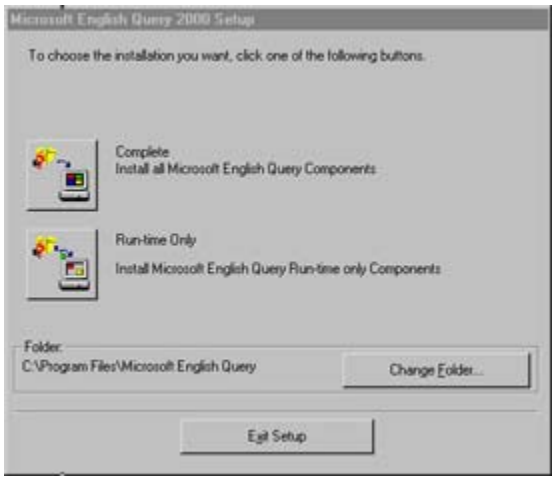


Figure 8.3: English Query Setup.

7. After the installation has completed, click OK to exit the English Query Setup program.

Note Although English Query is included with Microsoft SQL Server 2000, it is not dependent on SQL Server 2000. English Query can be installed on any compatible computer for developing or running English Query applications that connect to several different types of databases.

Installation Requirements

English Query can be installed on a computer running any of the following operating systems:

- Windows 95
- Windows 98
- Windows NT version 4.0 (Service Pack 6 or later)
- Windows 2000

The computer must have at least 40MB of disk space available. English Query also requires Microsoft Internet Explorer version 5.0 or later.

Creating an English Query Application

Creating an English Query application involves planning, testing, and implementing your solution. With the regression features and the new wizards in English Query 2000, testing and implementation are much easier than designing and planning your solution. In developing a robust and useful English Query application, it is really important to know what the users want the application to do and the vocabulary that they will use to interact with it.

Planning Your English Query Application

As with developing any application, the first thing that you should do is gather the requirements for your English Query application. What type of information does the user want? What type of questions will the users ask? Is there any special vocabulary that the users will need the application to understand? Once you know what your English Query application is supposed to understand and do, building your English Query solution is the easy part.

You should develop a test plan from your requirements for your English Query application *before* you start developing your English Query application. In particular, your test plan should include a list of questions that your English Query application should be able to answer from users, in the vernacular in which the users would ask the question. Having this plan early on will help ensure that your application is capable of doing everything it needs to do.

The other thing that you should do before you start your English Query application is to make sure that the database entities that you will query are as normalized as they can be. If you aren't creating the tables specifically for the English Query application and you can't normalize the tables, you should create normalized views of the tables that you are not allowed to normalize. Normalizing the database will make it much easier for the wizards to do the bulk of your development, plus it will make the English Query run-time engine more effective because it was designed to work with normalized databases. This is true for two reasons. One, a well-normalized database is easy to model (for both you and the wizards) with simple atomic relationships. Second, a series of simple atomic relationships that are derived from well-normalized database are easier for the English Query engine to process.

Understanding Users' Questions

To understand users' questions, English Query must have an understanding about what database entities can be queried, how they are related, the users' vernacular, and how that vernacular relates to the database entities. English query takes two steps in understanding the users' questions:

- Semantic analysis
- Semantic matching

The English Query run-time engine can analyze a natural-language sentence structure, figuring out the nouns and verbs and how they are related. However, without the knowledge you give it by building a semantic model, it does not know how to transform those nouns and verbs into database queries.

Note The English Query Wizards don't automatically include views when they make suggestions for entities and relationships. If you want to include views as entities in English Query Model, you must add them after you have completed the wizards.

Once the English Query run-time engine realizes what the nouns and verbs are in a sentence, it tries to match them to entities in the database. To allow the engine to be able to do this semantic matching, you must provide a model that contains all the semantic information about the entities and relationships in the database. The most obvious piece of semantic information that is needed to understand a user's questions are the English names that correspond with the database entities. For example, we would need to define the name of the Customer table in the Northwind database as *customers* for English Query to be able to recognize it in English statements. You can start to see how an English Query such as *list all customers* would translate into a database query.

What if the marketing department calls customers *clients* , and customer service calls them *customers* ? English Query uses synonyms to provide alternate names for database entities such as *clients, purchasers, and buyers* . After you define *clients* as a synonym for the Customer table, *list all clients* would provide the same results that were generated for *list all customers* . You can define several words to correspond to an entity so that different users are allowed to use their language and terminology to query the database and not be bound to a few specific terms.

This example works, but it is relatively simple. How about the natural-language query *list all clients that have sales* ? By performing a semantic analysis, we can easily see that there are two nouns, *clients* and *sales* . You might even have the information that links *clients* and *sales* to the Customer and Orders tables, but how are you going to be able to generate a database query that models all of this information? In order for English Query to be able to transform this statement into a database query, you need to create a semantic *relationship* between the two entities. A semantic relationship consists of an English phrase that the English Query run-time engine can match and translate into a database join or relationship. With this knowledge defined in your English Query semantic model, the English Query engine is able to create SQL statements that combine multiple tables, providing answers to complex statements such as this example.

Creating an English Query Project

After you perform an analysis of the users' needs, you will be ready to create an English Query project. Creating an English Query project requires the coordination of several files, but due to many user interface enhancements, this process has been made much easier. The foundation of user interface enhancements is the Microsoft English Query Development environment that uses InterDev to edit the English Query projects and files.

English Query Project Components

Many elements come together to form an English Query project, each having its own role in delivering your English Query solution:

- English Query Project (*.eqp)
- English Query Module (*.eqm)
- English Query Regression (*.eqr)
- English Query Domain (*.eqd)

English Query Project (.eqp), Module (*.eqm),* and *Regression (*.eqr)* files are all written in XML 2.0, and they can be opened with tools such as standard XML editors, Microsoft Notepad, or Microsoft Internet Explorer. The English Query Project contains the project name, version, database information, defaults, and a listing of all the English Query Module and Regression files in the project. In the Explorer view of the Microsoft English Query Development environment, shown in Figure 8.4 , the project and SQL Connection represent the information contained in the English Query Project file. English Query Modules are the core of your English Query development because they contain all the semantic information that is used to translate a natural English question into a database query. In addition, it contains the database schema that allows you to work with your English Query project in the absence of a database connection. The English Query Regression file contains the natural-language queries, their restatements, and SQL equivalents using your model. These files can be used to regression test your model over time to ensure consistent results.

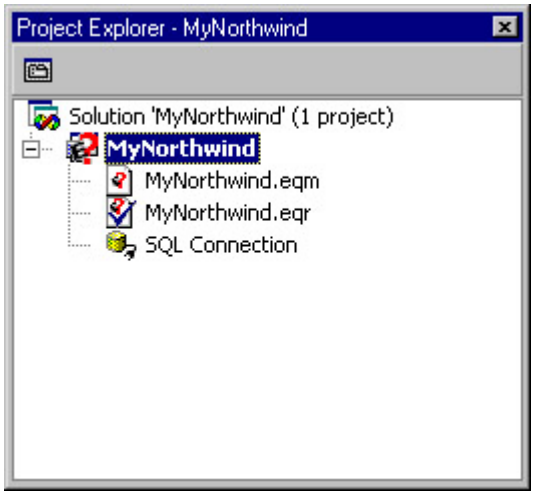


Figure 8.4: The Project Explorer view showing the English Query Project Module and Regression files.

The *English Query Domain (*.eqd) file* is a compiled binary file that is the output from your English Query project and modules. The English Query Domain file is used by the English Query run-time engine to transform natural English questions into database queries. An English Query Domain file is compiled for a very specific version of the

English Query run-time engine. If you upgrade the English Query engine, you must recompile the project for that specific version.

Here's how to inspect the contents of an English Query Module:

1. Right-click an English Query Module (*.eqm) file icon. A pop-up menu should appear.
2. Go down to Open With and select Choose Program.
3. In the Open With dialog box, select Internet Explorer.

English Query SQL Project

Microsoft English Query offers two different ways to start your English Query project: with an empty project or using one of the Project Wizards to automatically create a base project from your database model. Some complex database models or application requirements might be outside the scope of English Query's Project Wizards, and the need to define all your entities and relationships manually will be the only way to create your solution.

To create an empty English Query project and manually define all entities, relationships, and project properties, use the following steps:

1. Under the File menu in the Microsoft English Query development tool, select New Project. You will see the New Project dialog box.
2. In the New Project dialog box, select the Empty Project icon and enter a project name in the Name field **.**
3. Under the Project menu, choose the <Project Name> Properties menu item.
4. In the Project Properties dialog box, click the Change button on the SQL Connection frame.
5. In the Data Link Properties dialog box, shown in Figure 8.5 , select the data source type you need to connect to, and then click the Next button.



Figure 8.5: The OLE DB Provider Properties tab.

6. On the Data Link Properties Connection tab, shown in Figure 8.6 , enter the connection information for your SQL Server database. Select the OK button to continue.

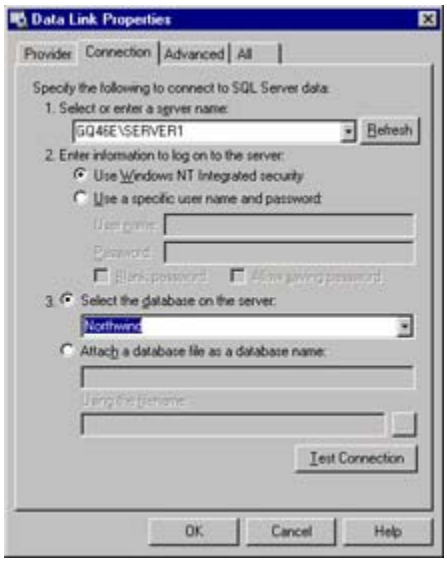


Figure 8.6: The Data Link Connection Properties tab.

7. Add each entity and relationship to your project as detailed in the "Creating and Editing Entities" and "Creating and Editing Relationships" sections that follow.

The second and seemingly more popular method for starting your English Query project is by using one of the Project Wizards included with Microsoft English Query 2000. The SQL Project Wizard will interrogate the database schema and suggest a set of entities for your tables, columns, and relationships, leaving you with a basic model that you can continue to develop and modify in order to complete your solution.

Creating an English Query SQL Project Using the SQL Project Wizard

1. Under the File menu, select New Project. You will see the New Project dialog box, as shown in Figure 8.7 .

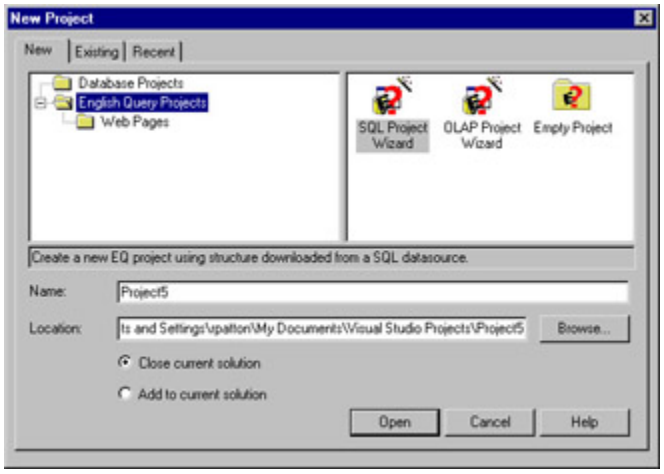


Figure 8.7: The New Project Dialog box.

2. In the New Project dialog box, select the SQL Project Wizard icon and enter a project name in the Name field.
3. In the Data Link Properties dialog box (refer back to Figure 8.5), select the data source type you need to connect to, and then click the Next button.
4. On the Data Link Properties Connection tab (refer back to Figure 8.6), enter the connection information for your SQL Server database. Select the OK button to continue.
5. The next dialog box displays a list of tables and views that are defined in the database you specified in Step 4. Select the tables and views that you want to include in your English Query project from the Available list and add them to the Selected list. After you have added all the tables and views you want to use, select the OK button to continue.
6. The Project Wizard **,** illustrated in Figure 8.8 , dialog will display each entity that was derived from the tables and views you selected in the previous step. Click the symbol next to each entity to display the suggested relationships that you can choose to accept or decline. After you review the entities and relationships that English Query has generated for your project, click the OK button to continue working with your English Query solution.

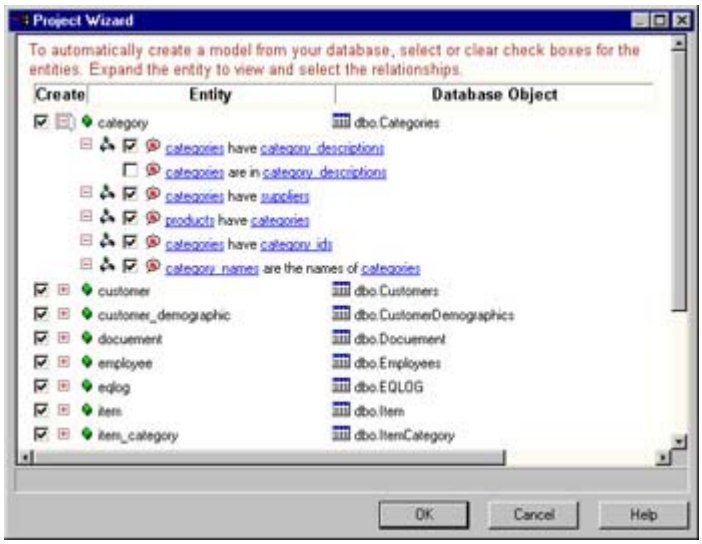


Figure 8.8: The English Query Project Wizard.

English Query OLAP Project

Creating an English Query OLAP project is much like creating a SQL Project; you can do it without the wizard or with the wizard. If you elect to create a project without the wizard, you must add the entities and relationships one by one.

These are the steps for creating an English Query OLAP project from scratch:

1. Under the File menu in the Microsoft English Query development tool, select New Project. You will see the New Project dialog box (refer back to Figure 8.7).
2. In the New Project dialog box, select the Empty Project icon and enter a project name in the Name field.
3. Under the Project menu, choose the <Project Name> Properties item.
4. In the Project Properties dialog box, click the Change button on the SQL Connection frame.
5. In the Data Link Properties dialog box (refer back to Figure 8.5), select the data source type you need to connect to, and then click the Next button.
6. On the Data Link Properties Connection tab (refer back to Figure 8.6), enter the connection information for your SQL Server database. Select the OK button to continue.
7. Add each entity and relationship to your project, as detailed in the “Creating and Editing Entities” and “Creating and Editing Relationships” sections that follow.

Alternatively, you can use the new OLAP Project Wizard. Like the SQL Project Wizard, this wizard interrogates the cubes and suggests a set of entities and relationships, leaving you with a basic model at the completion of the wizard steps.

These step are for creating an English Query OLAP project using the OLAP Project Wizard:

1. Under the File menu, select New Project . You will see the New Project dialog box (refer back to Figure 8.7).
2. In the New Project dialog box, type in a name and then select OLAP Project Wizard.
3. In the Select Analysis Sever dialog box, select the analysis server and database that you want to connect to, and then click the OK button.
4. A dialog box with the available OLAP Cubes will come up. Select the ones you want, and then click the OK button.
5. Now you will see the project wizard with an entity from the OLAP objects (Dimensions, Level, Fact table, etc.) in the OLAP Cubes you chose. If you click the symbol next to each entity, you will see suggested relationships that you can choose to accept or decline. After you review the relationships for each entity, click the OK button.

The English Query Model

An *English Query Model* is all the semantic information describing the database entities that your application will be able to query. This includes database schema and the entities and relationships that you build using the English Query development environment.

Creating and Editing Entities

After you have created your model, you might need to edit existing entities or add new entities to your model. You can edit or add a new entity in your English Query Model with the Entity/New Entity dialog box, as shown in Figure 8.9 You can add a new entity by right-clicking Semantic Objects in the Semantics tab, and then click Add Entity in the pop-up menu. If you want to edit an existing entity, you need to double-click the entity in the Semantics tab. Once you have the Entity/New Entity dialog box open, you can enter or modify the Database and Semantic properties that define that entity.

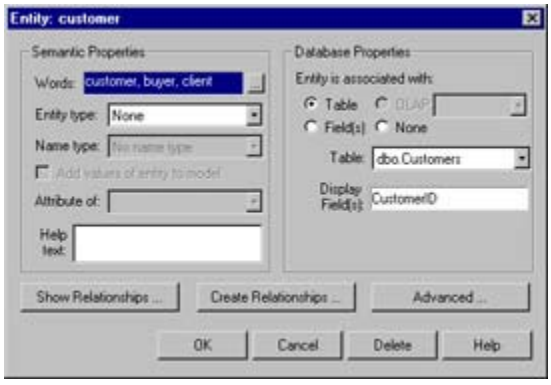


Figure 8.9: The Entity dialog box for customer.

Typically, when we define an entity, we start with the database because a well-designed database is very easily thought of in terms of entities. The *Database properties* are the fields that let you define a table, field, OLAP level, dimension, measure, property, or fact as an entity. You choose whether you want to define a table, field, or part of an OLAP

cube, and then you choose the specific instance from the combo boxes listing those that are available.

Now that you have defined or edited the database information for the entity, you need to define the semantic properties that will allow this database entity to be understood in terms of natural English-language expressions. One of the most important semantic properties for the entity is the *Words* property. These are the natural-language words that identity the entity. If you wanted to add a synonym for the entity, you would add the singular form to the list of words. *Entity type* identifies the type of words that describe the entity: Are they *Who, Where, When,* or a *Measure* ? *Name type* is a bit different from the Entity type in that it is available only for fields, levels, and entities. It tells us the type of name of the entity: Proper, Common, or Classifier. *Attribute of* is used to signify that this entity is an attribute of another entity and provides a list of available entities to choose from.

To add an entity:

1. In the model editor, right-click Semantic Objects on the Semantics tab, and click Add Entity in the pop-up menu.
2. In the New Entity dialog box, enter the singular form of the word that identifies the entity into the Words box. Press Enter.
3. For each synonym (a synonym can be a word or phrase) of the word that identifies the entity, enter its singulars form into the Words box. Press Enter.
4. In the Database Properties frame, choose OLAP as the value for the type of database entity. The words you have entered in the Words box should be associated with the Entity.
5. Enter the specifics for the type of OLAP that defines what the Entity is associated with.
6. In the Entity Type combo box, select from a list of enumerated types.
7. If you selected OLAP levels or OLAP properties as the database entity that this English Query entity is associated with, you can choose a Name Type from the combo box that has a list of enumerated types.
8. Click the OK button to finish.

Creating and Editing Relationships

Relationships in your English Query project define the links between the entities that you have defined. These links are important for allowing the English Query engine to decipher complex English statements that request related information from multiple tables in your database. An example of this is a user asking the system to “List all customers with total sales over $500.” The relationship Semantic Object provides you with the ability to tie entities together. The Relationship/New Relationship dialog box, as shown in Figure 8.10 , is used to establish or modify these relationships between entities. You can add a new relationship by right-clicking Semantic Objects in the Semantics tab, and then clicking Add Relationship in the pop-up menu. To edit existing relationships, double-click the relationship in the Semantics tab. Once you access the Relationship/New Relationship dialog box, you can add entities and phrasing for those entities.



Figure 8.10: The Customers_have_employees Relationship dialog box.

The upper-left area of the Relationship/New Relationship dialog box contains a list box of the entities that participate in the selected relationship. Using the buttons to the right, you can add, remove and edit the entities in the Relationship. The Add button allows you to add the phrasings that are used to identify the type of phrases people use that are indicative of this relationship. The Remove button allows you discard a phrasing that might not be useful. For instance, workers might have once spoken about a relationship in these terms, but they do not anymore. The Edit button allows you to define a role that the entity plays in this relationship.

The lower-left area of the Relationships/New Relationships dialog box contains a listing of the phrases that have been defined for the entity relationship. Phrasings are the ways that the entities are discussed in natural language. For example, “customers have contact names” is a phrasing that is an English representation of a relationship between two entities. Again, with the buttons to the left-hand side, you can Add, Edit, and Remove phrasings. When you click Add, you are brought to a new screen where you choose the type of semantic structure for your phrasing. Once you choose the type of phrasing, you are brought to dialog box specific to that phrasing. You go to that same dialog box, the one specific to the type of phrasing, when you click Edit. You cannot change the type of phrasing by clicking Edit. If you have created a relationship with the wrong type of phrasing, you must delete it and create a new one with the right type of phrasing.

To add a relationship:

1. In the model editor, right-click Semantic Objects on the Semantics tab, and click Add Relationship in the pop-up menu.
2. In the New Relationship dialog box, click the Add button to the left of the Entities list.
3. In the Select Entities dialog box, choose the entities that you want to participate in the relationship, and click the OK button.

4. In the New Relationship dialog box, click the Add button to the left of the Phrasings list.
5. In the Select Phrasings dialog box, select the Phrasing type and click the OK button. In the phrasing dialog box specific to the phrasing type, fill out the semantic information that is required, and then click the OK button.
6. Click the OK button to finish.

Building and Deploying Your English Query Application

To deploy your English Query application, you need to compile the English Query Model (the *.eqd file). You can do this by choosing the Build item (Ctrl + Shift + B) on the Build Menu in the project editor. Once you have compiled the English Query application, you can deploy it as part of many different types of applications, including:

- Microsoft Visual Basic
- Microsoft Visual C++
- Microsoft Internet Information Server (IIS)

No matter what development platform you are using to create your client application interface, you must distribute the dynamic link libraries (DLLs) that make up the English Query run-time engine and the English Query Model for your project.

Implementing Web-Based English Query Applications

Many English Query applications use a Web-based user interface, so English Query's Two-Click Deployment feature offers an efficient method for deploying your English Query solution. To take advantage of two-click deployment, you need to have the correct rights on the Web server; you must be a member of the operators groups, and you must have permission to write to the root of the Web server on which you are choosing to create your project. In addition, you must have Visual InterDev installed. If you do not have InterDev installed, the Deploy to Web menu item is disabled and inaccessible.

To implement a Web-Based English Query Application:

1. Select Deploy to Web from the Project menu to display the Web Project Wizard.
2. The Web Project Wizard allows you to specify the Web Server you want to deploy to and the Mode you want to author in. After you have answered those questions, click the Next button to continue.
3. The next dialog box asks you to specify which Web site to add your application to. You can choose an existing site or create an entirely new Web site. If you choose an existing Web site, click the Finish button to continue. Otherwise, you need to click the Next button and answer two more questions about your new Web site.
4. You will be asked to choose a layout for your Web site. You can choose one from the list or navigate to the one that you want to use. When you are satisfied, click the Next button to continue.
5. Finally, you will be asked to choose a theme. Once again, you can choose one from a list of templates or navigate to the one that you want to use. When you are satisfied with the theme, click the Finish button.

Testing Your English Query Application

The Microsoft English Query development environment now includes the *Model Test window.* The Model Test window, shown in Figure 8.11 , allows you to thoroughly test the English Query Model by allowing you to enter questions and then translating your English question into:

- A natural-language restatement of the question
- A natural-language answer to the question
- A query (in SQL or MDX) generated from the question
- Results of the query against the database

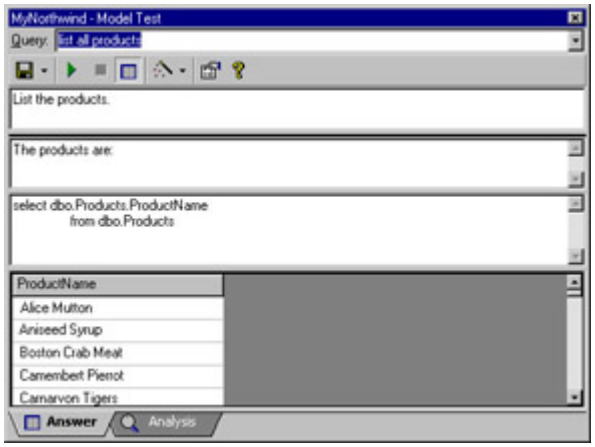


Figure 8.11: Testing an English Query Model with the Model Test application.

With this interactive tool, you can perform extensive *unit testing* against your English Query Model. You can also save the results of your unit testing for *regression testing* . Each English Query Solution can contain Regression files (*.eqr) in an XML format that allows you to store the following values from the unit tests in the Model Test application:

- The question
- The restatement
- The answer
- The query

It won't save the data, because data changes over time and the data is not really the output that matters from English Query. The output that matters is the query and the natural-language statements. During the lifetime of your application, you can run the regression files to ensure that you don't break part of your system as you build others.

To activate the Model Testing window, click the Debug menu and select the Start or Start Without Debugging menu items. You can also start Debugging mode from the keyboard using the F 5 or Ctrl + F 5 keys. Once the Model Testing window is active, you can enter your questions and see the results. If you would like to add any of your interrogations from the session, click the Save icon, and that will add the question and its results to a regression file, as shown in Figure 8.12 . Once you have a regression file you approve of, you can run the regression file, compare the results against previous expected output, and

promote the output. You would promote the output if the tests results differ from the old file but are more correct.

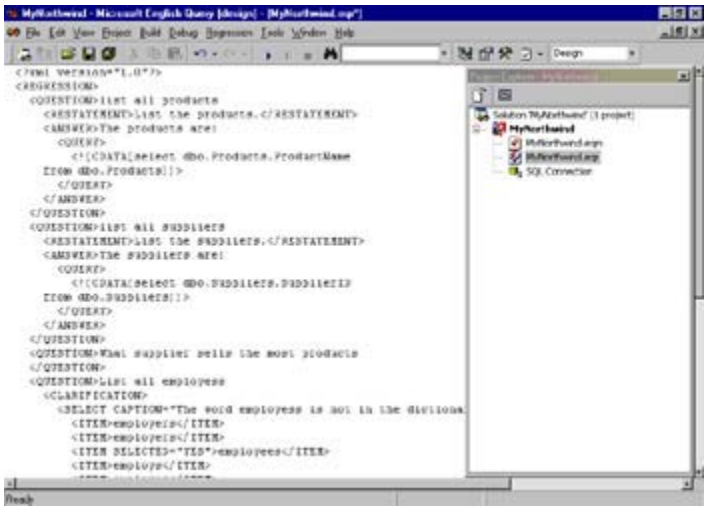


Figure 8.12: Regression text XML in the English Query editor.

You should test the English Query Model before compiling it into an application to make sure that the questions the users are likely to ask are supported by the model. Afterward you should modify it according to the deficiencies brought out in your test. After you are satisfied with the performance of the model, build it into a compiled English Query application, known as an English Query Domain (*.eqd file).

The Suggestion Wizard is available within the Model Test window and is useful during the testing phase. You can start by asking a question that users are likely to ask, and the Suggestion Wizard will present suggested entities and relationships needed to answer the question.

Tip Running the generated query against the database is not the default option. In order to see the results of the SQL query against the database, you must click the View Results button or use the Ctrl + R shortcut.

Putting It All Together

You have learned about all the components of Microsoft English Query 2000. You have learned how to build, test, and deploy your English Query solutions. Now you are ready to use all this information to create a fully functional English Query application. Refer to the following section to build and deploy an English Query solution using the Northwind sample database.

Creating a Web-Based English Query Solution

1. Select the Microsoft English Query icon from the Programs | Microsoft SQL Server | English Query program group to start the English Query development environment.

2. From the File menu, select New Project to display the New Project dialog box that was shown in Figure 8.7 .
3. In the New Project dialog box, enter **NorthwindWeb** in the Name field and select SQL Project Wizard.
4. In the Data Link Properties dialog box, select Microsoft OLE DB Provider for SQL Server, and click the Next button to continue.
5. Enter the name of the server that you want to connect to and the logon information for that server. Choose Northwind as the database, and then click the OK button to continue.
6. A dialog box with the available database tables and views will come up. Select all of them (>>) and then click the OK button.
7. You will see the Project Wizard with an entity corresponding to each table you chose. Review the relationships for each entity, and then click the OK button. You will see a new English Query Solution with all the needed files; the solution is named NorthwindWeb and has the correct extension attached.
8. Choose Build under the Build menu.
9. Choose Deploy to Web under the Project menu.
10. When the Web Project Wizard appears, you need to specify the Web server you want to deploy to and Master as the mode you want to author in. After you have answered those questions, click the Next button.
11. Now the Web Project Wizard will ask you to specify which Web site to put your application in. Choose “Create new Web application” and leave the name as NorthwindWeb. Check “Create search.htm to enable full-text searching” and then click the Next button.
12. Choose the layout Left 1, then click the Next button.
13. Finally, choose Leaves as the theme, and click the Finish button.
14. On completion, you should have a complete Web site like the one shown in Figure 8.13 , able to interact with the database using natural language.



Figure 8.13: A functioning Web site that can interact with the user via natural language.

An Overview of Full-Text Search

Previous SQL Server developers were limited to the realm of simple pattern searching to mine the volumes of character data stored in database servers. Sometimes we would find what we were looking for, but often we would not get all the results, or they were not quite what we wanted. Then Microsoft introduced SQL Server 7, which included, for the first time, full-text searching. That feature opened up enormous power to do smart, "fuzzy" types of searches—searches that would return the same results for *hammered* as for *hammering* , for *airplane!* as for *Airplanes;* searches that could put entries that have both *white* and *paper* at the top of the list when the user queried for *White Paper* , but would also include the records that had only *white* or *paper* .

File Filtering

Perhaps the most important addition to full-text searching and indexing in SQL Server 2000 is the addition of *file filtering* . File filtering is the ability to search through binary documents stored in SQL Server using the FREETEXT and CONTAINS predicates. You can search through many types of documents stored in an image column in your SQL Server database, including:

- Microsoft Word documents (*.doc)
- Microsoft Excel documents (*.xls)
- Power Point presentations (*.ppt)
- Text documents (*.txt)
- HTML documents (*.htm)
- Any type of document that you write a filter for; you can write your own filters for different file types using the Microsoft Platform Software Development Kit (SDK)

To be able to place a full-text index on an image column that contains documents, another column in that table must contain the extension for the document. This column is known as a *binding column* . The binding column can contain any kind of character data and can contain blank and NULL values; if there is no extension in the binding column, it assumed that the document is a text file. Documents in an image column do not have to be of the same type. In fact, you can store many different kinds of documents in a single column. However, the maximum size of a document in the image column is 16MB, and filtered text inside the document cannot be more than 256KB.

The ability to index external documents in SQL Server 2000 is provided by Microsoft Search Service. The Microsoft Search Service *filters* the documents in a full-text index, stripping the text out of a document and storing the filtered text in the full-text index. Once it is in the full-text index, the text is as searchable as any other character data stored in the database.

Full-Text Search Architecture

As shown in Figure 8.14 , the Microsoft Search Service provides, through an OLE DB interface, SQL Server full-text searches. All the data for the full-text indexes are stored outside SQL Server, and all searching is done through Microsoft Search Service, as opposed to the SQL Server Query engine. When SQL Server's Query engine determines it needs to perform a full-text search, it accesses the Microsoft Search Service. Microsoft Search Service then executes the query and passes back the results to the query engine.

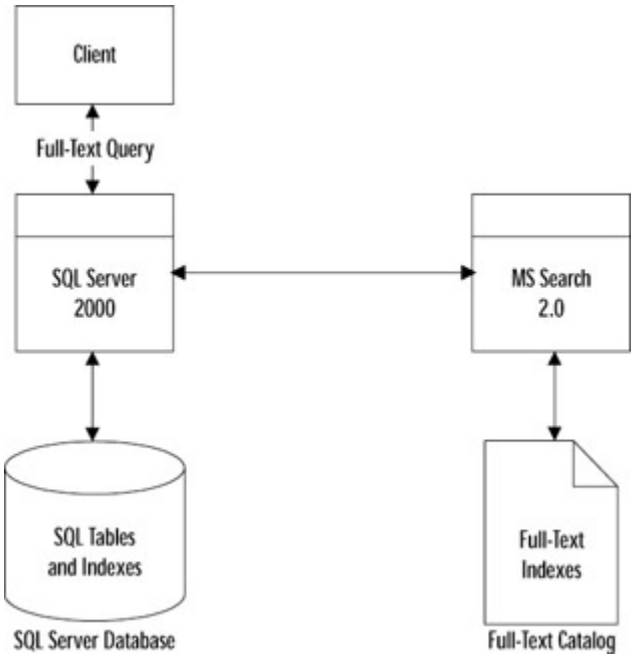


Figure 8.14: The full-text search architecture.

In addition to performing the searches, the Microsoft Search Service also maintains, populates, and builds the full-text indexes. The Microsoft Search Service maintains a series of files that comprise the full-text indexes in *full-text catalogs*. Many indexes can be in one catalog, but they must all be from the same database, because a catalog cannot span databases. Even so, their number is limited per server, not database; you are limited to a maximum of 256 full-text catalogs per server. Due to the fact that SQL Server does not maintain full-text indexes, they do not behave like SQL Server indexes. A SQL Server index is always and immediately updated to reflect changes that happen to the data they represent, whereas, in most cases, a full-text index must be explicitly rebuilt to reflect any changes to the data.

Microsoft Search Service

Microsoft Search Service is an external service to SQL Server that must be installed in order to perform full-text searches. Microsoft Search Service is a component that is installed during a typical installation. Although you can have many SQL Server instances running on a single computer, you can have only one instance of the Microsoft Search Service on any computer.

The Microsoft Search Service is external to SQL Server, and it will run in the context of the local system account. However, the account running SQL Server must be an administrator of the Microsoft Search Service. When you install the Microsoft Search Service, this relationship will be set up correctly. To ensure that this relationship is maintained correctly, you should change the account running SQL Server only through the Properties tab of the SQL Server Properties in Enterprise Manager. If you change it elsewhere, such as through the services interface in the control panels, the correct changes will not propagate through to the Microsoft Search Service.

Performance Considerations for Full-Text Indexes

You can do several things to get the best performance out of your full-text searches. You can configure the hardware in an optimal way, tune the queries, and set operating parameters. One of the most basic things you can do is ensure that you have the required hardware to obtain acceptable performance. Microsoft recommends the following configuration:

- At least one CPU; 500MHz Xeon III processors or better
- At least 1GB RAM
- Separate disk controllers and channels for the full-text indexes and for the page file
- RAID0+1 or RAID5 for the page file (mirroring)
- RAID0, RAID0+1, or RAID5 for the full-text indexes

This type of configuration is needed when you have large full-text indexes (over a million rows) or when your systems rely on high performance from your full-text searches. If you do not depend on your full-text searches to perform to such a degree, you can run them with the minimum recommended hardware for your SQL Server.

As with most systems, the better processors, memory, and disks can make for a better-performing system. The number of processors, their speed, and the amount of random access memory are important to full-text indexing when population is occurring, because populating a full-text catalog is very processor and memory intensive. The searching itself is not very processor intensive. Even with a very powerful system, the majority of populations should be done in off hours. If you elect to use the background population option, make sure that your system's hardware is robust enough to handle it and that the table is important enough to justify such attention.

However, most database applications are not bottlenecked due to a lack of processing power or physical memory. Most systems are disk-IO bound. This is very important to consider when you are designing your full-text indexing system, because you can do certain things to reduce the risk of a potential bottleneck. If you have the resources, you should place both the virtual memory file and the full-text catalogs on arrays of disks separate from the ones that contain the files for the rest of the database. The page file needs to be on a set of drives that provide fast read/write access, such as RAID0 and RAID0+1. Although it is good idea to place full-text catalogs on disk arrays with fault

tolerance, they do not have to be placed on such an array, because all the data is *derived* from data stored on other drives. If the disk drive expires where the full-text catalogs resides, you can rebuild them with no loss of data. In addition to being on their own disks, both the page file and the full-text catalogs should have their IO channels and, if possible, their own IO controller. Please refer to Table 8.1 for common RAID levels. Paging files should be on high-speed disk subsystems, but RAID5 fault tolerance isn't necessary.

Table 8.1: Common RAID Levels

Level	**Description**	**Fault Tolerance**	**Read Performance**	**Write Performance**
RAID0	Striping without parity	No	Fast	Fast
RAID1	Mirroring	Yes	Normal	Normal
RAID5	Striping with parity	Yes	Fast	Slower
RAID0+1	Striping with mirroring	Yes	Fast	Fast

A performance concern in SQL Server 7 was the fact that you could have only eight active full-text rowsets open at any given time. If you had more than eight, they would be placed into a waiting queue, and often queries would start timing out without being executed. This caused problems in systems that received a great number of full-text searches and, to make matters worse, this parameter was hard-coded into MSSearch Service 1.0. This obstacle has been removed with SQL Server 2000, thanks to MSSearch Service 2.0, which has a default setting of 1024 concurrent connections.

In addition to a much larger initial value, you can use the sp_fulltext_service system stored procedure with the resource_usage parameter to adjust with the number of concurrent connections allowed. You can set it to any of these values:

- 128
- 512
- 1024
- 2048
- 4096

If you wanted to increase the number of concurrent connections to 2048, you would run this statement in the query analyzer:

```
EXEC sp_full_text_service 'resource_usage', 4
```

SQL Books Online describes how to use this stored procedure with the resource_usage parameter, but it indicates only that you can set it at 1–5: 1 corresponds to 128, 2 to 512, 3 to 1024, 4 to 2048, and 5 to 4096.

Due to some performance issues with very large tables (tables with over a million rows) that have full-text indexes, Microsoft introduced in SQL Server 7 Service Pack 2 the optional parameter Top_N_By_Rank for the CONTAINSTABLE and FREETEXTTABLE predicates. If you put 100 as the Top_N_By_Rank parameter, 100 rows will be returned. This system decreases the amount of communication SQL Server had with the Search Service, but it does not decrease the amount of work that the Search Service must do. The Search Service does an index scan each and every time it runs a query.

One thing to note is that the Top_N_Rank parameter is applied before any conditions in the WHERE clause. So, if you execute a query seeking the top 100 items from a full-text catalog, the top 100 would be returned from the Microsoft Search Service. If you had a WHERE clause filtering on a status field, the WHERE clause would be applied after SQL Server receives the results from the Search Service, so your query would often return fewer than 100 results. This concept can be illustrated by writing a query that does a full-text search for the products that match *the hottest Sauce!* and are sold by SupplierID 2. First, we would write a query without the WHERE clause and the Top_N_Rank parameter:

```
SELECT     FT_TBL.[ProductName], KEY_TBL.RANK, FT_TBL.SupplierID
FROM       Products AS FT_TBL
           INNER JOIN
           FREETEXTTABLE(Products, [ProductName], 'the hottest Sauce!')
AS KEY_TBL
           ON FT_TBL.[ProductID] = KEY_TBL.[KEY]
ORDER BY KEY_TBL.RANK DESC
```

And you would get the following results:

```
ProductName                          RANK      SupplierID
-----------------------------------  -----     -----------
Louisiana Fiery Hot Pepper Sauce      80       2
Northwoods Cranberry Sauce            40       3
Louisiana Hot Spiced Okra             40       2
(3 row(s) affected)
```

As you can see, three rows would be returned, and two of them would have the SupplierID equal to 2. If we were to add the WHERE clause to filter by SupplierID, the query would look like this:

```
SELECT     FT_TBL.[ProductName], KEY_TBL.RANK, FT_TBL.SupplierID
FROM       Products AS FT_TBL
```

```
           INNER JOIN
           FREETEXTTABLE(Products, [ProductName], 'the hottest Sauce!')
AS KEY_TBL
           ON FT_TBL.[ProductID] = KEY_TBL.[KEY]
WHERE FT_TBL.SupplierID = 2
ORDER BY KEY_TBL.RANK DESC
```

And we would get this as the result:

```
ProductName                          RANK      SupplierID
-----------------------------------  -----     ----------
Louisiana Fiery Hot Pepper Sauce      80       2
Louisiana Hot Spiced Okra             40       2
(2 row(s) affected)
```

Everything seems as though it is going well. We have two rows, as we would expect, but if you add a Top_N_Rank parameter with a value of 2, like this:

```
SELECT     FT_TBL.[ProductName], KEY_TBL.RANK, FT_TBL.SupplierID
FROM       Products AS FT_TBL
           INNER JOIN
           FREETEXTTABLE(Products, [ProductName], 'the hottest
Sauce!',2)
AS KEY_TBL
           ON FT_TBL.[ProductID] = KEY_TBL.[KEY]
WHERE FT_TBL.SupplierID = 2
ORDER BY KEY_TBL.RANK DESC
```

You would get this:

```
ProductName                          RANK      SupplierID
-----------------------------------  -----     ----------
Louisiana Fiery Hot Pepper Sauce     80        2
(1 row(s) affected)
```

Now it returns only one row, even though you know that are two rows that will return TRUE for *the hottest sauce* and that have SupplierID values of 2. What happened was that the Top_N_Rank parameter filters *before* the WHERE clause because FREETEXTTABLE and CONTAINSTABLE return tables that participate in a join. The table that FREETEXTTABLE would have returned can be illustrated with this query:

```
SELECT     FT_TBL.[ProductName], KEY_TBL.RANK, FT_TBL.SupplierID
FROM       Products AS FT_TBL
           INNER JOIN
           FREETEXTTABLE(Products, [ProductName], 'the hottest
Sauce!',2)
AS KEY_TBL
           ON FT_TBL.[ProductID] = KEY_TBL.[KEY]
ORDER BY KEY_TBL.RANK DESC
```

Now you can see that it would have returned a table with two records, as directed by the Top_N_Rank parameter. These results are now joined back to the Products table, and only one of them has a SupplierID value of 2, which would leave use with one record:

```
ProductName                              RANK      SupplierID
---------------------------------        -----     ----------
Louisiana Fiery Hot Pepper Sauce         80        2
Northwoods Cranberry Sauce               40        3
(2 row(s) affected)
```

Enabling Full-Text Search

Although the components that provide full-text search capabilities to SQL Server 2000 are external to SQL Server, they are set up during a typical installation of SQL Server Standard and Enterprise editions. However, you must be running a version of Windows 2000 or Windows NT to have it installed; The Microsoft Search Service will not install on computers running Windows 98 or Windows 95.

Creating a Full-Text Catalog

Full-text catalogs are the vessels that contain full-text indexes. All the data that are contained within a full-text index are in the full-text catalogs. A catalog can contain many indexes from different tables, but all those tables must be in the same database as the catalog itself. A catalog cannot span multiple databases. A catalog can be created with a number of tools:

- Query Analyzer
- Enterprise Manager
- Full-Text Indexing Wizard

With the Query Analyzer, in the database in which you want to create that catalog, execute a statement like this:

```
EXEC sp_fulltext_catalog 'CatalogName',' create'
```

After executing this statement, you will have created a catalog named CatalogName in the default full-text catalog directory. If you wanted to do the exact same thing in Enterprise Manager, you would navigate over to the Catalogs under the database and server you wanted, and then right-click on the details pane. Select the first item, New Full-Text Catalog, in the pop-up menu, and type **CatalogName** in the name text box. After you clicked the OK button, you would have a new full-text catalog. As mentioned, you could create a full-text catalog through the Full-Text Indexing Wizard, but you would have to create an entire full-text index at the same time. The following steps show you how to create a full-text catalog with this wizard.

Enabling a Database for Full-Text Search

1. In Enterprise Manager, select that database on the server on which you want to enable full-text indexing.
2. Click the Wizard icon on the toolbar or select the Wizards menu option from the Tools menu to display the Select Wizard dialog box.
3. Select the Full-Text Indexing Wizard option from the Database group, and click the OK button to continue.
4. When the Full-Text Indexing Wizard comes up, it will prompt you for the table you want to index. After you select the correct one, click the Next button.
5. Choose the unique index that will be used that table, and click the Next button.

6. Choose the character field(s) that you want to be part of a full-text index, and then click the Next button.
7. The wizard will ask you which catalog to place the index in (and give you the ability to create a new one). When you have specified one, click the Next button.
8. Finally, the wizard will allow you to select or create a population schedule. When you have specified one, click the Next button.
9. Click the Finish button.

As you can tell, we did quite a bit more than just enable the database for a full-text search. We also enabled a table for full-text searching, and we built a full-text index on that table. If we had wanted to enable only the database for a full-text search, we could have done that with the Query Analyzer, like this:

```
USE Northwind
EXEC sp_Full_Text_database 'enable'
```

sp_full_text_database takes 1 parameter, action, which is a varchar string that can be either *enable* or *disable* . Enable enables full-text-searches on the database; disable removes this ability.

Enabling a Table for Full-Text Search

If your database is already enabled for a full-text search but you would like to enable another table for a full-text search, you can do that with Enterprise Manager:

1. In Enterprise Manager, select the table in the database on the server on which you want to enable full-text indexing.
2. Click the Wizard icon on the toolbar and select the Full-Text Indexing Wizard or select the Wizards menu option from the Tools menu to display the Select Wizard dialog box.
3. When the Full-Text Indexing Wizard comes up, choose the unique index that will be used for that table, and click the Next button.
4. Choose the character field(s) that you want to be part of a full-text index, and then click the Next button.
5. The wizard will ask you which catalog to place the index in (and give you the ability to create a new one). When you have specified one, click the Next button.
6. Finally, the wizard will allow you to select or create a schedule for the catalog that you are using. When you have specified one, click the Next button.
7. Click the Finish button.

This process is very similar to enabling full-text searching for the entire database. The only difference is that the wizard did not need to prompt you for that table, because you already had it selected in Enterprise Manager. If the database was not enabled for full-text searching, it is now. If you had wanted to enable a table for full-text searching without creating a full-text index, we could have done that with the Query Analyzer, like this:

```
EXEC sp_full_text_table 'TableName', 'create', 'CatalogName',
  'UniqueIndexName'
```

Enabling a Column for Full-Text Search

1. In Enterprise Manager, select the table in the database on the server on which you want to enable full-text indexing.
2. Click the Wizard icon on the toolbar, and select the Full-Text Indexing Wizard or select the Wizards menu option from the Tools menu to display the Select Wizard dialog box.
3. If there is no full-text index on the table already, the wizard will prompt you for the unique index for the table. Choose the unique index that will be used for that table, and click the Next button.
4. If there is a full-text index on the table, you would have advanced straight to this step. Choose the character field(s) that you want to be part of a full-text index, and then click the Next button.
5. The wizard will ask you which catalog to place the index in (and give you the ability to create a new one). When you have specified one, click the Next button.
6. Finally, the wizard will allow you to select or create a schedule for the catalog that you are using. When you have specified one, click the Next button.
7. Click the Finish button.

Once again, if you prefer, you could add a field to an existing full-text index, like this:

```
Use northwind
EXEC sp_full_text_column TableName, ColumnName, 'add'
```

Tip You can activate the Full-Text Wizard from the Tools menu by selecting the Full-Text Indexing item in all of the cases we just examined, instead of selecting the Wizard icon. In addition, you could have brought up the wizard by right-clicking the icon for the table you wanted to enable, then clicking Full-Text Index Table. If there were already an index on the table, you would choose Edit Full-Text Indexing…; otherwise, you would select Define Full-Text Indexing on a Table.

Creating a Full-Text Index on the Products Table in the Northwind Database

1. Open SQL Server Enterprise Manager and drill down to the Tables in the Northwind database using the tree view.
2. Right-click the Products table in the design pane.
3. Select the Full-Text Index Table, item and then select “Define Full-Text Indexing on a Table …” on the context menu.
4. You will see the splash screen of the Full-Text Indexing Wizard. After you read it, click the Next button.
5. Select PK_Products as the unique index for the query processor, and click the Next button.
6. Select ProductName as the only character field you want to participate in the index. Click the Next button.

7. Click the “Create a new catalog” check box, and type **ProductsCatalog** in the text box to the right of Name. Click the Next button.
8. Click the New Table Schedule button.
9. In the box to the right of Name, type **ProductsIncremental** , and select Incremental Population under the Job Type. Then select Recurring under Schedule Frequency. Click the Change button.
10. Choose Daily under Occurs, and then click the OK button.
11. Click the OK button on the schedule screen, and then click the Next button in the wizard.
12. Review your selections. If you are satisfied, click Finish.
13. Click OK in the confirmation dialog box.

If you try querying this index immediately, you will be in for a surprise, because no rows will come back. In order to receive results when you query it, you must first do a full population of the index.

Building the Full-Text Index

By building the catalog and the index, you have built the vessels that can contain the data, but you have not yet placed the data in the vessels. You can query the tables, and SQL Server will not generate an error, but you will not get any results.

To populate a catalog:

1. In Enterprise Manager, open the server and database that your catalog is in, and click Full-Text Catalogs.
2. In the pane to the right, right-click the catalog that you want to populate under full-text catalogs in your database on your server.
3. On the pop-up menu, click Start Full Population.
4. Click the OK button in the confirmation dialog box.

You will be able to query the table while it is populating and receive results based on how far that population has progressed. Population is memory and CPU intensive, so your database might not respond as well as usual during this time period. You can monitor the progress of your population using FullTextCatalogProperty in the query analyzer. This feature can provide you with quite a bit of information concerning your population, such as the overall status and the number of items in the index. Here are the statements you could run to monitor the population status of the Items ProductsCatalog catalog:

```
WHILE (FullTextCatalogProperty('ProductsCatalog', 'populatestatus') <>
0)
BEGIN
       WAITFOR DELAY '00:00:01'
       Print 'Item Count:' +
       Cast(FullTextCatalogProperty('ProductsCatalog', 'itemcount') as
       Varchar(10))
```

```
        Print 'Catalog Size:' +
        Cast(FullTextCatalogProperty('ProductsCatalog', 'indexsize') as
        Varchar(10)) + ' MB'
END



Print 'FINISHED'
Print '----------'
Print 'Item Count:' + Cast(FullTextCatalogProperty('ProductsCatalog',
'itemcount') as Varchar(10))
Print 'Catalog Size:' + Cast(FullTextCatalogProperty('ProductsCatalog',
'indexsize') as Varchar(10)) + ' MB'
```

Tip During periods of population, you might want to set SQL Server's maximum memory to half the RAM available on the machine, to allow more memory to be used for the population. In addition, you might want to increase the amount of resources dedicated to the Search Service using the sp_full_text_service procedure with the resource_usage parameter. Of course, you should reset these setting to their original values when the population is completed.

When an entry is added to an index during a population, it is scanned and parsed into its base terms, removing punctuation and "noise words." *Noise words* are words such as *the, because,* and *only* that are so common that they can be considered "background noise." It is often a good idea to add noise words specific to your line of business, to prevent these words from giving you inaccurate results. How to add noise words:

1. Locate your noise words file for your server. It is located in the full-text data config directory of your SQL instance installation—for example, the directory:
   2. `C:\Program Files\Microsoft SQL Server SERVER1 FTDATA\SQLServer$[ServerInstance]\Config\`

   Noise files are built for specific languages, so if we are doing our searches on data that are in English, we must edit noise.eng. If we wanted to edit the Japanese noise file, we would chose noise.jpn.

3. Right-click the noise file icon, and choose the Open With menu item.
4. Select Notepad from the list of applications.
5. Enter the words that you want to add to the noise file. Press Enter.
6. Open the Services Administrative control panel.
7. Right-click the Microsoft Search service icon, and select the Stop menu item.
8. Choose the Save menu item under the File menu.
9. Right-click the Microsoft Search service, and select the Start menu item.
10. Repopulate each of your catalogs.

Querying Full-Text Indexes

Four predicates were added to transact SQL in order to allow users to perform full-text searches:

- FREETEXT
- FREETEXTTABLE
- CONTAINS
- CONTAINSTABLE

CONTAINS and FREETEXT are filtering conditions that limit what is returned in your query, as would a join. CONTAINSTABLE and FREETEXTTABLE each return a table containing a KEY and RANK to which your query must join.

FREETEXT and FREETEXTTABLE

FREETEXT is the simplest full-text predicate to use. It requires only two parameters: the field that you are searching and the search string that you are looking for. It returns TRUE if finds a match for the values in the search string within the columns you are searching. This query in the Northwind database:

```
SELECT     ProductID, ProductName
FROM       Products
WHERE      FREETEXT (ProductName,'The Hottest Sauce!')
```

It will return the following if it had a full-text index on ProductName:

```
ProductID     ProductName
---------     ----------------------------------
66            Louisiana Hot Spiced Okra
65            Louisiana Fiery Hot Pepper Sauce
8             Northwoods Cranberry Sauce
(3 row(s) affected)
```

ProductName was the column that we searched through on the full-text index, and our search string was *The Hottest Sauce!* . You will notice that two of the results contain the word *Sauce* , making it obvious why they would have returned TRUE. However, the product with ProductID 66 has none of the terms *The, Hottest,* or *Sauce* in its name, but it does have *Hot* , and that is why it was selected.

Tip If a full-text index has more than one column that is part of the index, you can search through all the columns by placing an asterisk (*) as the column parameter in any of the full-text predicates. The results will be based on data in all the columns that are part of the index.

FREETEXT searches try to match the meaning of the search string by factoring out inflection, superlatives, and gender. When a term is submitted with the FREETEXT

predicate, the search string immediately removes all the noise words and punctuation from it. In the case of the *The Hottest Sauce!* , *The* and the exclamation mark would have been removed, to leave you with *Hottest Sauce* . Then the search string is parsed into individual terms *Hottest* and *Sauce* . The words would then be broken down in to their base terms, removing plurality, inflection, suffixes, and tense, to leave us with the search terms *Hot* and *Sauce* . Finally, the system would scan the index looking for these base terms in the column that we selected. Knowing how this works, you can now see how you would have achieved the same results looking for *Hotter Sauces* or *The Sauce Hot?* as we did looking for *The Hottest Sauce!* .

Looking at our example, you might say that it is great, but shouldn't the Louisiana Fiery Hot Pepper Sauce be the first item in the results? After all, it contains both *Hot* and *Sauce* . However, FREETEXT doesn't offer any way to rank the data; it only returns TRUE or FALSE.

That is where FREETEXTTABLE comes in. FREETEXTTABLE returns a table with two columns, RANK and KEY. RANK is the relative score of the search string in the column(s) in the FREETEXT query, and KEY is the unique field that you identified in the setup of the full-text index. With KEY, we are able to join back to our base table, and with RANK, we can provide a relative score of the results. If we wanted to list items from the product table ordered most likely to least likely matches for the *The Hottest Sauce!* , then, we could do so with this query.

```
SELECT     KEY_TBL.RANK, FT_TBL.[ProductID], FT_TBL.[ProductName]
FROM      Products AS FT_TBL
          INNER JOIN
          FREETEXTTABLE(Products, [ProductName], 'the hottest Sauce!')
          AS KEY_TBL
          ON FT_TBL.[ProductID] = KEY_TBL.[KEY]
ORDER BY KEY_TBL.RANK DESC
```

Our results:

```
RANK        ProductID     ProductName
-------     ---------     ----------------------------------------
80          65            Louisiana Fiery Hot Pepper Sauce
40          66            Louisiana Hot Spiced Okra
40          8             Northwoods Cranberry Sauce
(3 row(s) affected)
```

RANK is relative score from 0 to 1000; it is not a percentage. A higher RANK value means only that the item is believed to be a better match than an item with a lower value. You might think that if we entered a product name exactly as it was listed in the database, we would get a perfect score. If you thought that, you would be wrong! Look at this example, which searches for the product Northwoods Cranberry Sauce:

```
SELECT     KEY_TBL.RANK, FT_TBL.[ProductID], FT_TBL.[ProductName]
FROM       Products AS FT_TBL
           INNER JOIN
```

```
           FREETEXTTABLE(Products, [ProductName], 'Northwoods Cranberry
           Sauce') AS KEY_TBL
           ON FT_TBL.[ProductID] = KEY_TBL.[KEY]
ORDER BY KEY_TBL.RANK DESC
```

It returns:

```
RANK        ProductID     ProductName
-------     ---------     --------------------------------
90          8             Northwoods Cranberry Sauce
17          65            Louisiana Fiery Hot Pepper Sauce
(2 row(s) affected)
```

RANK is generated from a sophisticated proprietary algorithm that uses these factors:

- The total number of rows that contain the parsed term
- The total number of times the parsed term occurs in a column
- The total number of unique word occurrences in the table

How they are balanced to come up with RANK is not disclosed, and it really isn't important. What is important is that RANK provides a relevance ranking within the context of the query.

CONTAINS and CONTAINSTABLE

CONTAINS is similar to FREETEXT with respect to the fact that it only returns TRUE or FALSE, but it differs with respect to how much power it gives you to shape the values that will return TRUE. You can search for a specific phrase, prefixes to a phrase, terms that are near each other, or different inflections of the same term. We could use the following query to look for the phrase *the hottest sauce!* using the CONTAINS predicate.

```
SELECT     [ProductID], [ProductName]
FROM       [Products]
WHERE      CONTAINS ([ProductName],'"the hottest Sauce!"')
```

We would get this result:

```
ProductID     ProductName
---------     --------------------------------
(0 row(s) affected)
```

The CONTAINS predicate is looking for a simple phrase and is behaving a lot like an old-fashioned query using the LIKE operator. However, there are two benefits to using this method over the LIKE operator when you have a full-text index:

- Punctuation filtering
- Performance

Full-text queries are much more efficient that those using LIKE. As you add more and more rows, full-text searches continue to become faster and faster, compared with LIKE. Furthermore, the CONTAINS predicate will filter punctuation marks off the beginning and end of the search phrase when looking for a match. Therefore, *Cranberry Sauce* is equivalent to *!Cranberry Sauce, Cranberry Sauce!!* , *Cranberry Sauce* , and *!!Cranberry Sauce!.* Searching for a prefix to a phrase is also very much like using a query with LIKE:

```
SELECT     [ProductID], [ProductName]
FROM       [Products]
WHERE      CONTAINS ([ProductName],'"Cran*"')
```

It returns:

```
ProductID     ProductName
---------     ----------------------------
8             Northwoods Cranberry Sauce
(1 row(s) affected)
```

In addition to being able to do things similar to the LIKE operator, the CONTAINS predicate can also do things like the FREETEXT predicate. For instance, it can look for different inflections of the same word:

```
SELECT     [ProductID], [ProductName]
FROM       [Products]
WHERE      CONTAINS ([ProductName],' FORMSOF (INFLECTIONAL, Hottest)')
```

This returns:

```
ProductID     ProductName
---------     ---------------------------------
66            Louisiana Hot Spiced Okra
65            Louisiana Fiery Hot Pepper Sauce
(2 row(s) affected)
```

If they are so similar, why would we ever use CONTAINS instead of FREETEXT? CONTAINS offers a number of different options, including the Boolean operators, weighting of terms, and proximity searches. These different options can be used in varying combinations, allowing you pinpoint control in your searches. To see an example of the control CONTAINS offers you, you could write a query that would return all the products with any variation of *hot* , but the result must also have an exact match with the word *sauce* . So you could use this query:

```
SELECT     [ProductID], [ProductName]
FROM       [Products]
WHERE      CONTAINS ([ProductName], '"Sauce" AND FORMSOF (INFLECTIONAL,
           Hottest)')
```

It returns:

```
ProductID      ProductName
----------     ----------------------------------
65             Louisiana Fiery Hot Pepper Sauce
(1 row(s) affected)
```

CONTAINSTABLE is to CONTAINS as FREETEXTTABLE is to FREETEXT. Instead of returning a TRUE or FALSE, CONTAINSTABLE returns a KAY and a RANK. One of the options that the CONTAINS predicate provides is the ability to weight terms in the results. That ability did not really do CONTAINS very much good, because it is only interested in TRUE or FALSE and is not concerned with the degree of the match. However, the ability to weight terms is very important to CONTAINSTABLE. It allows you to perform searches in which you can apply knowledge that you have. For instance, you could be looking for *Hot Sauce* but might realize that you are more likely to find what you are looking for if you find a match for *Sauce* as opposed to *Hot* . You could construct a CONTAINSTABLE search that embodies that idea, like this:

```
SELECT     KEY_TBL.RANK, FT_TBL.[ProductID], FT_TBL.[ProductName]
FROM       Products AS FT_TBL
           INNER JOIN
           CONTAINSTABLE(Products, [ProductName], 'ISABOUT(hot weight
           (.2), sauce weight (.8))') AS KEY_TBL
           ON FT_TBL.[ProductID] = KEY_TBL.[KEY]
ORDER BY KEY_TBL.RANK DESC
```

This would give you the following results:

```
RANK      ProductID     ProductName
-----     ---------     ---------------------------------------
75        65            Louisiana Fiery Hot Pepper Sauce
59        8             Northwoods Cranberry Sauce
14        66            Louisiana Hot Spiced Okra
(3 row(s) affected)
```

If we were more interested in *Hot* than *Sauce,* we could switch the weights, which would alter the order of our results:

```
SELECT     KEY_TBL.RANK, FT_TBL.[ProductID], FT_TBL.[ProductName]
FROM       Products AS FT_TBL
           INNER JOIN
           CONTAINSTABLE(Products, [ProductName], 'ISABOUT(hot weight
           (.8), sauce weight (.2))') AS KEY_TBL
           ON FT_TBL.[ProductID] = KEY_TBL.[KEY]
ORDER BY KEY_TBL.RANK DESC
```

This change would return the results in this order:

```
RANK      ProductID     ProductName
-----     ---------     ---------------------------------
75        65            Louisiana Fiery Hot Pepper Sauce
59        66            Louisiana Hot Spiced Okra
14        8             Northwoods Cranberry Sauce
(3 row(s) affected)
```

Administering Full-Text Catalogs and Indexes

No matter how carefully crafted or swiftly it is executed, a query is only as valuable as the data that it can return. If significant data have changed and a full-text index has not been, the results of your query's return might not be very useful. Knowing this, it is obvious that the most important task concerning the administration of full-text catalogs and indexes is maintaining the data in the catalogs through backups and population.

You can inspect most of the important statistics regarding the administration of a full-text catalog by looking at the Full-Text Catalog Properties dialog box, shown in Figure 8.15 . It includes the number of items, the unique key count, and the last population date. You can bring up this dialog box by right-clicking the catalog in Enterprise Manager and choosing the Properties item from the pop-up menu.



Figure 8.15: The Full-Text Catalog Properties dialog box.

Backing Up Full-Text Catalogs

As stated earlier, full-text searching is a service provided by Microsoft Search Service and is external to SQL Server. So, when you back up a database, you are not backing up all the pieces that are needed for full-text searching. The good news is that the system tables are backed up, and the system tables hold information that defines the full-text catalogs. These are the pieces of information that SQL Server uses to interface with the Microsoft Search Service. The bad news is that the full-text catalogs themselves, with the data, are not backed up when you back up a database.

Tip In order to minimize the amount of resources used on your front-line servers, you can back up your databases to secondary servers and populate the full-text catalogs on them. Once the population is complete, you can back up the full-text catalogs from the secondary servers and restore back to your front-line servers.

You don't have to back up the full-text catalogs. All the data in the full-text catalogs/indexes are derived data, and if there was an accident, you could always rebuild them from the source data in the SQL Server. However, repopulating these files could be very time intensive, depending on the amount of data you have in them.

Whether you back up the catalogs or not, you must make sure that the paths for the full-text catalogs on the server that you are backing up exist on the server to which you are restoring the database. You can check the paths of your full-text indexes by running the following statement in the Query Analyzer:

```
EXEC   sp_help_fulltext_catalogs
```

The paths of the full-text catalogs will be listed in the PATH column when you run this stored procedure. If NULL is the value, then the catalog is in the default location, MSSQL7\FtData. Using this stored procedure, you can verify that all the paths exist. If they do, you can rebuild and repopulate the catalogs after you restore the database to that machine. If you used some other software to backup the full-text catalogs, you could place the backups of the catalogs in the correct directories now, but there could be some problems if you are not careful.

To back up full-text indexes:

1. Open the Services Administrative control panel.
2. Right-click the Microsoft Search Service, and select the Stop menu item.
3. Use backup software to back up all the files to the backup device.
4. Right-click the Microsoft Search Service, and select the Start menu item.

To restore from backup:

1. Open the Services Administrative control panel.
2. Right-click the Microsoft Search Service, and select the Stop menu item.
3. Ensure that the paths on which the full-text catalog files are supposed to exist do exist on the server you are restoring.
4. Put the files in the right path.
5. Make sure that the database ID and the table IDs match.
6. Use backup software to back up all the files to the backup device.
7. Right-click the Microsoft Search Service, and select the Start menu item.

The primary obstacle that you have to overcome in restoring full-text catalog files is that the location of the files for the full-text catalog is not a value in the database; it is the *value of the database* . If you look at the directories in the MSSQL7\FtData directory, you'll notice several folders with cryptic names. These names have a very specific meaning and are in the form SQL0000{dbid}0000{fcatid}, where dbid is the database ID and ftcatid is the full-text catalog ID. If you have restored onto a new server, the database ID could be different from the database ID on the old server. This will throw everything out of sync. By detaching and adding databases, you need to make sure that you restore to the same database ID that you had when you backed up the files.

Populating Full-Text Indexes

Population is requested on either a catalog-by-catalog or table-by-table basis. Populating on a catalog basis allows you to populate multiple indexes in one operation; populating tables lets you populate specific indexes. Full-text indexes are different from SQL server table indexes in two respects. First, full-text indexes are not automatically maintained when data change; second, you do not rebuild a full-text index but instead rebuild a full-text catalog, which contains the full-text indexes. There are three methods to populate your full-text indexes:

- Full population
- Incremental population
- Change-tracking population

A *full population* repopulates the entire index, reading each and every row. Typically, you do a full population after you initially build the index, and then maintain the index with either incremental populations or change-tracking populations.

An *incremental population* updates the rows since the last time that the index was populated. In order to perform an incremental population on a table, the full-text index requires a timestamp field. Each time a full or incremental population is executed, the time is stored so that it can be compared against the column with the timestamp value. Any row that has been modified since the last rebuild becomes part of the incremental population. A timestamp field is updated each time any field in the rows is updated, regardless of whether that field is part of the full-text index or not, unless the table modification was performed with UPDATETEXT or WRITETEXT statements. UPDATETEXT or WRITETEXT are unlogged operations and will not automatically update the timestamp field; if the timestamp field is not updated, the index will not be updated. You can try to execute an incremental population on a table without a timestamp field, but the result will be a full population because the system cannot determine which rows have been updated.

New in SQL Server 2000 is the ability to maintain full-text indexes with *change tracking* . When you activate change tracking, a log is maintained in a system table, and updates to the table are propagated to the index from the changes.

To enable change tracking for a table, you use the sp_fulltext_table stored procedure, like this:

```
EXEC   sp_fulltext_table 'TableName', 'Start_change_tracking'
```

Once you have activated change tracking, you can propagate the changes to the indexes continuously, on demand, or on a schedule. Maintaining your indexes continuously is called a *background population* because it is always going on in the background. Because it is always executing, it is very CPU and memory intensive. If you have an index that is

very important to keep up to date and you have the available resources, you can start a background population from the Query Analyzer, like this:

```
EXEC   sp_fulltext_table 'TableName', 'Start_change_tracking'
EXEC   sp_fulltext_table 'TableName', 'start_background_updateindex'
```

If you cannot continuously dedicate the resources to maintain a particular full-text index in the background, you can still take advantage of the new change-tracking option. You can use it on demand or on a schedule. At certain periods of time, usually the wee hours of the night, your database servers have resources to spare due to a scarcity of usage. When you know that you have available resources, you can use the sp_fulltext_table procedure with the update_index value in the action parameter to force the changes to propagate. You can do this in the Query Analyzer, like this:

```
EXEC   sp_fulltext_table 'TableName', 'update_index'
```

Scheduling Index Rebuilds

As mentioned before, most SQL Servers have memory and CPU to spare during very predictable periods of time, when user activity is low. You can schedule repopulations of the indexes during those hours. SQL Server Agent can be instructed to schedule an on-demand update by telling it to execute the sp_fulltext_table stored procedure with the update_index parameter and the correct table name. If you want to take advantage of these wee hours to keep your indexes up to date with full and incremental populations of your catalogs, you can also create a job for SQL Server Agent to execute the sp_fulltext_catalog values start_incremental or start_full for the action parameter. However, it is easier to do it through Enterprise Manager. If you navigate Enterprise Manager to the Catalogs icon under the database and server that you are using, you will see all the catalogs that have been created in your database. If you right-click the catalog icon, a pop-up menu will come up with Schedules as an item. If you select Schedules on the menu, a dialog box will appear, listing all the scheduled populations for that catalog, as shown in Figure 8.16 .

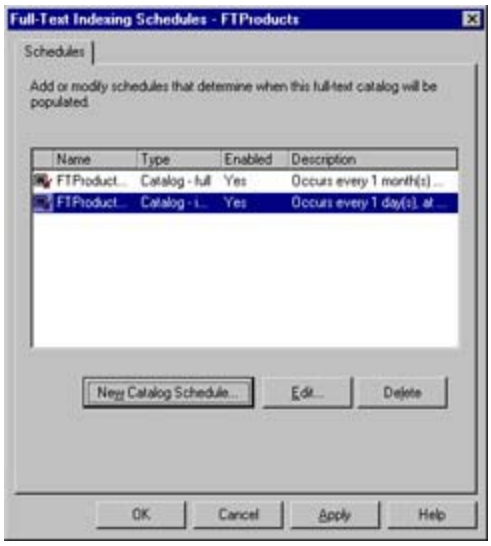


Figure 8.16: Full-Text Indexing Schedules for a catalog.

You can create, delete, or remove any schedule for that catalog. If you click the New Catalog Schedule… button, the dialog box in Figure 8.17 will come up. This box allows you to choose the type of population, either full or incremental, and the frequency of its execution. Once you complete choosing the parameters that you desire, a job is created for the SQL Server Agent. You can edit this schedule with the Agent interface or with the Full-Text Indexing Catalog Schedule interface.

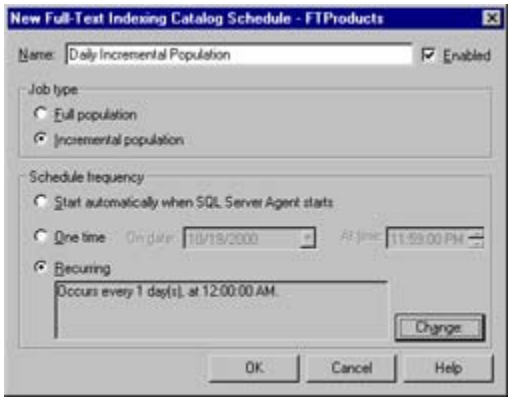


Figure 8.17: The New Full-Text Indexing Catalog Schedule dialog box.

Summary

English Query and Full-Text Searching are external pieces of software that open up new ways of working with the databases in SQL Server. English Query liberates users from having to know SQL or MDX. Full-text searches open up efficient, sophisticated searches through text data that don't require as exact matches and can rank their results in relative order. In addition to searching through text data in the database, full-text searching can also scour binary documents that are stored within the database.

English Query is a series of COM objects and development tools that are completely external to SQL Server 2000. As you can see in Figure 8.18 , SQL Server 2000 does not even directly interface with English Query. Instead, the English Query run-time engine is a translation engine that client applications can use as a translator between natural language and SQL or MDX. This feature allows the user to ask such questions as "How many customers have bought hot sauce?" and receive the correct answer, without the user knowing anything about the database structure or SQL. English Query has been around for a few years, but it has been hindered because it was difficult to work with. Through a number of new features with the release of English Query 2000, its shortcomings have been eliminated. It is now easy to build, test, maintain, and deploy English Query applications.

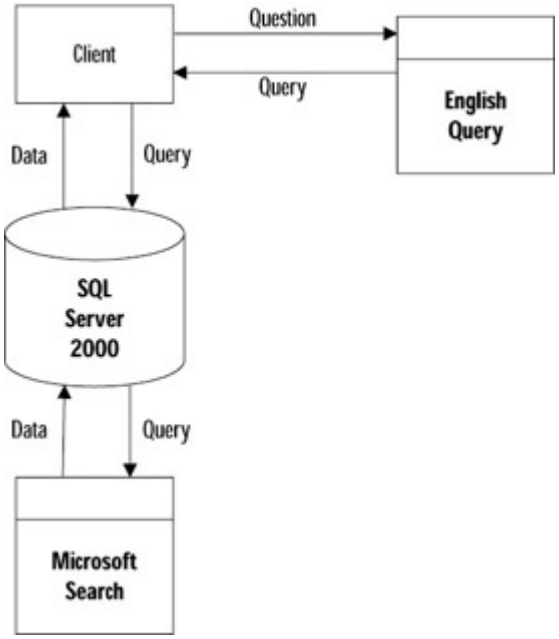


Figure 8.18: The SQL Server 2000 English Query full-text architecture.

Full-text searches aren't so much part of SQL Server as a service provided by interfacing with Microsoft Search 2.0. The Search Service is external to SQL Server, but unlike the English Query, it has built-in interfaces to SQL Server 2000 for searching and management. The interfaces to Microsoft Search allow users to search through text data in columns and documents in a variety of ways using SQL statements. Full-text search depends on SQL Server for indexing and searching text and image data.

FAQs

Q. Can I write an English Query application for a Windows 98 PC that uses Analysis Services?

A. For the most part, this is possible—with one caveat. The COM objects that comprise English Query itself run fine on operating systems such as Windows 95 and Windows 98. However, Microsoft SQL Server 2000 Analysis Services will not install on Windows 98, so you will need to install the SQL Server version 7.0 OLAP Services client on the computer. This will provide your application with most, but not all, of the functionality that was available with Analysis Services.

Q. I would like to have my full-text indexes perform the absolute best that they can. Is there anything else I can do?

A. If you really want your indexes to perform as fast as possible, you could store them on a solid-state drive. A solid-state drive is essentially a drive made of RAM that offers you a hundred times better read/write performance than the fastest RAID arrays. Like most computer products, these drives have come down in price quite a bit in the last few years.

Q. I have a full-text catalog with several tables, but I want to populate only one of them. Can you populate a single table and not an entire catalog?

A. Yes. You can choose to populate either an entire catalog that would have several tables or a single table. However, you must initiate the population of the table from the Query Analyzer as opposed to Enterprise Manager. Using the sp_fulltext_table system stored procedure, you could start a full population of the products table, like this:

```
EXEC  sp_fulltext_table 'Table_Name', 'start_full'
```

In the same manner, you could perform an incremental population or a background population, respectively, like this:

```
EXEC  sp_fulltext_table 'Table_Name', 'start_incremental
EXEC  sp_fulltext_table 'Table_Name', 'update_index'
```

Q. I have a database server that is heavily used for full-text searches. How should I organize the primary files in my database system to minimize the risk of being disk-IO bound?

A. Figure 8.19 shows an example of a potential system. There are three disk arrays: one for the virtual memory, one for the database files, and one for the full-text catalog. Each has its own channel to help with throughput. The disk arrays for the virtual memory array and for the database files are both RAID5, so a disk can fail and the system will continue to function. The full-text catalog is on a RAID0 disk array proving good performance but no fault tolerance. If one of the disks failed, your database would continue to function, but you would not be able to perform full-text searches until you rebuilt the catalogs on a functioning disk array. In addition to the location that you place the virtual memory, database, and full-text files, you should also consider in which drive you are going to place your transaction logs and operating system files (discussed in more detail in Chapter 14).

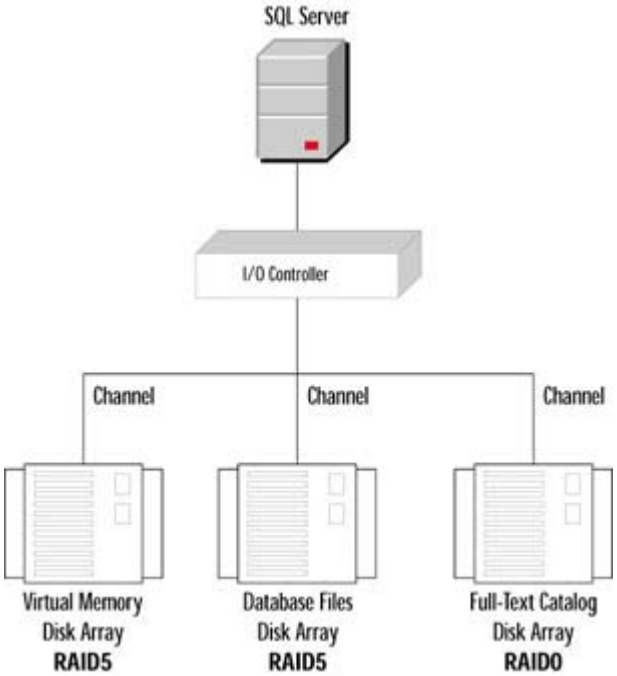


Figure 8.19: A potential disk configuration.

Q. How would I implement a search for my Web site that brought back results in order from best match to worst match? Which predicate would I use?

A. Since you want to have your tables ranked, you need to use one of the TABLE predicates:

- FREETEXTTABLE
- CONTAINSTABLE

Both return a table that includes the unique identifier that you can join back to your original table and a field that contains the RANK of the row for the text that you are searching for. You can order your results by the RANK parameter, and that will give you your best match to least match. If you do not want to tweak your results to weight them based on different terms, use FREETEXTTABLE. If you need to weight the search terms, you need to use CONTAINSTABLE.

Chapter 9: Importing and Exporting Data

Introduction

Transferring and manipulating data was once the job of trained database administrators. With data the driving factor in markets that require reaction time in terms of minutes, having the information available to analyze is the key to making informed and accurate decisions for your organization. Getting data in and out of SQL Server is a common task, and several tools are available in SQL Server 2000 to accomplish this task. Whether it's importing data from heterogeneous systems such other databases or exporting data for use in other database or user applications such as OLAP solutions, Microsoft SQL Server can get the job done with as little as a few clicks of the mouse.

Tools such as SQL Server's Data Transformation Services (DTS) have become such important components in database solutions that it is difficult to remember how these tasks were accomplished before they were available. SQL Server 2000 has made several enhancements to DTS, allowing access to tasks such as file transfers using FTP and nested package execution. Additional enhancements such as saving DTS packages to Visual Basic files and global variables make DTS a powerful programming tool for importing, exporting, and manipulating SQL Server data.

Traditional data import/export utilities are also available in SQL Server 2000. Bulk Copy (BCP), a longtime favorite of database administrators, is supported, as is T-SQL's BULK INSERT command. This chapter reviews each of these import/export methods and their advantages and disadvantages. You will learn when to use one method rather than another and work through examples of implementing each utility to accomplish your import, export, and data manipulation tasks.

Overview of Data Import and Export Tools

One of the more common tasks involved with administering a database is importing and exporting data. The necessity to import and export data can be spawned by a request from a user, an upgrade to a system, data consolidation, archiving, or testing new application technology. Just as numerous as the reasons to move the data are the number of native and heterogeneous formats in which the data reside. Sometimes the data can reside in another database, such as Microsoft Access or Oracle. Other times, the data source is not in a database and is encapsulated in a Microsoft Excel spreadsheet or a text file. Adding more complexity is the fact that the data can reside in various disparate locations; a needed file could be located on a network share on the LAN while other needed information is on a database server in another city on the WAN. Issues are further complicated because often the data cannot be used "as is" and need to be transformed to be useful. Compounding the difficulty is the fact that the data are typically "needed yesterday" and the import/export task typically needs to be repeated again and again. The endless combinations of data formats and locations can make the tasks of importing and exporting daunting.

Due to these factors, the task of importing and exporting data to and from SQL Server was formerly a grueling and time-consuming job that required a highly skilled database administrator. It did not matter how small the data set was; a great deal of work was involved to move data to and from the various data sources. Because of the complexity and importance of the task, Microsoft has made importing and exporting data has been made easier, faster, and more robust through an abundance of tools in the release of SQL Server 2000.

One of the most popular tools is Data Transformation Services (DTS). With DTS, you can create a package or multiple packages to transform, consolidate, and transfer data between various sources and destinations. The source or destination can range from a native SQL Server database to an IBM DB2 mainframe database or a text file. To allow such dynamic data interchange, DTS utilizes an array of OLE DB-compliant drivers. A DTS package is usually made up of one or more tasks. E-mail, FTP, Execute SQL, and the Data Mining task are only a small list of the available tasks that can be part of a package. The Execute Package task allows for nested execution of a package from within a package and is one of the new predefined tasks added in SQL Server 2000. The package can also contain custom tasks that have been added to DTS via the Register Custom Task command. From time to time, a task must execute before another task and execute only if the prior task completed successfully. The flow of tasks can be controlled using precedence constraints or ActiveX script, requiring a task to successfully complete before executing the next task.

In addition to DTS, SQL Server 2000 offers additional tools to import and export data. One popular tool is the classic Bulk Copy Program (BCP). BCP is a command-line utility that accepts a range of arguments and an optional format file to quickly transfer the data in and out of SQL Server. Over the years, database administrators and developers have written batch files to import and export data from a database using BCP. For the longest

time, BCP was the only tool that could be used to quickly and efficiently transfer large amounts of data in and out of a SQL Server database.

Another tool available to move data into and out of SQL Server is the SQL Server Distributed Management Objects (SQL-DMO). SQL-DMO can be referenced in Microsoft Visual Basic (VB) or Microsoft Visual C++ (VC++). SQL-DMO allows the user access to a multitude of predefined objects consisting of various properties, methods, and events. The objects within SQL-DMO are used to control or manage SQL Server and the various components of SQL Server. The BulkCopy and BulkCopy2 Objects within SQL-DMO work like the command-line BCP tool. They allow for the importing or exporting of large amounts of data to or from a data file and to or from a database table. A skilled database administrator or developer can programmatically control and automate the import/export of data VB or VC++. The use of a powerful programming environment like VB or VC++ can allow for the inception of business rules or conditions into the import/ export process.

Another tool is the T-SQL command BULK INSERT. BULK INSERT works like the command-line BCP tool, except that BULK INSERT can transfer only from a data file to a SQL Server database table. This gives BULK INSERT limited uses compared with the other available tools. The ability to execute and schedule this T-SQL command like a typical SQL query makes this tool a viable option for many applications.

Data Transformation Services

DTS is a set of components used to import, export, and transform data and the tools that let you use the components. The source and destination data can be from a single data source or multiple data sources. The data formats can vary greatly, from an Excel spreadsheet to a comma-separated text file or a relational database. Because DTS is a series of COM components, SQL Server does not have to participate by being either the source or the destination data source.

DTS allows for the simple creation of solutions to transfer, transform, and consolidate data. With a few clicks of the mouse and using the DTS Designer, a user can turn what was once a tedious data import or export task into a task that takes minimal effort. DTS has many programmable objects that can be used to develop and tailor import/export solutions to meet the needs at hand. With the many enhancements to DTS in SQL Server 2000, DTS is the tool with all the capabilities to develop any data import/export solution for any type of data transfer, transformation, or consolidation.

What's New in DTS?

A number of enhancements were added to DTS with the release of SQL Server 2000. Many of these enhancements were made to the tasks available in DTS. One of the more important enhancements to DTS was the new ability to add a COM component as a task external to DTS. This allows the user to create a custom COM component in Microsoft VB or Microsoft Visual C++ and add it as a task in DTS. In addition to providing the ability to use COM components, DTS has extended its capabilities by adding the ability to use Parameterized Source Queries in the Transform Data task and the Execute SQL task. The Execute Package task was improved by allowing it to dynamically assign the values of global variables in a parent package to a child package. In addition to enhancing the abilities of DTS tasks, these new tasks were added for your benefit:

- File Transfer Protocol (FTP)
- Execute Package
- Dynamic Properties
- Message Queue
- Parallel Data Pump Data
- Transfer Database
- Transfer Error Messages
- Transfer Logins
- Transfer Jobs
- Transfer Master Stored Procedures
- Analysis Services Processing
- Data Mining

Task improvement isn't the only area in which DTS has added features in this release. DTS packages can now be saved as Microsoft Visual Basic (.BAS) files. This feature opens up the world of programmatic development to DTS packages, allowing a large

pool of people who have the ability to work with and troubleshoot VB code to work with and troubleshoot DTS packages. Packages were also improved by adding an enhanced logging capability that allows the developer to save execution data for a DTS package. This information can be used to audit the execution history for the package by allowing the developer to view critical information about each process that ran in each task and package. The enhanced logging capability also allows exception files to be generated from the rows of data that failed to be processed due to error. Finally, a new Multi-Phase Data Pump allows users to handle and customize the data import or export tasks at various stages. In addition, the Multi-Phase Data Pump allows for row-level handling of data and the ability to customize initialization and termination steps when the pump starts and stops.

Note If the COM component is created using Microsoft Visual Basic, the task will execute only sequentially, not asynchronously, due to the lack of free threading in VB.

A Tool for Any Environment

Microsoft's DTS is a powerful tool that can be used for any environment that has a need to move data quickly and easily. It does not matter if the systems are Oracle, DB2, or Informix based; as long as you can connect to the database through an OLE DB driver, you can move the data effectively through DTS. SQL Server need not hold the destination or source data.

DTS's ability to move data between sources quickly and easily allows you to move data from a production transaction-based Oracle database server to a decision support based DB2 data warehouse. It allows you to collect data from Btrieve data stores and move them to Excel. DTS can leverage any data store that has an OLE DB driver.

Even though DTS does not need SQL Server to store data, it does need a licensed SQL Server in order for you to legally use the DTS. You can keep your data where it is, but you must add a SQL Server in order to utilize the power of DTS. In you do not already have a SQL Server, you can minimize the costs involved with leveraging the power of DTS by storing packages in the Structure Storage File format. This will ensure that you will not need additional client access licenses to communicate to the SQL Server.

Data Transformation Services Architecture

DTS is a series of COM components that allows you to move data between disparate sources, performing transformation consolidation and extraction as needed. Because it is a series of COM objects, DTS can be executed independent of SQL Server. A fundamental piece of DTS is the use of the OLE DB architecture. OLE DB was designed to communicate with a variety of data sources, and this ability allows DTS to achieve

communication with all sorts of heterogeneous data sources for which OLE DB drivers are available. Because of this dependence on OLE DB drivers, DTS is limited in what it can do with a specific data source by the functionally that the driver provides for that data source.

As you can see in Figure 9.1 , generally there is at least one source data connection and one destination data connection, although there can be multiples of both. Having multiple sources allows for consolidation, and having multiple destinations allows for extraction. In addition to having the ability to talk to the heterogeneous data, DTS provides the ability to transform and manipulate the data between data sources. All the information needed to communicate with, manipulate, and channel the data is encapsulated in the DTS package. The DTS package itself is made up of tasks, connections, the workflow between them, and the graphical user interface (GUI) layout information. The connections define the information needed to communicate with the data sources. Tasks are what is done to and with the data; they also encompass the other programmatic actions that are needed to be executed by the package. Workflow controls the order, sequence, and conditional flow of the tasks. The GUI layout is preserved in the package so that tools that access and edit the package can consistently render the package in the manner in which the user laid it out.

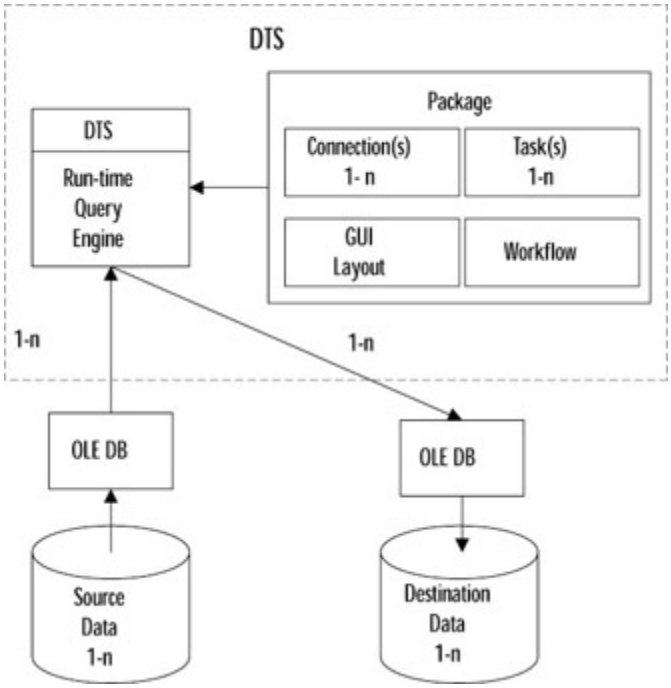


Figure 9.1: DTS architectural overview.

Packages

Much of the DTS architecture revolves around the *DTS package* , which is the container that is used to organize the steps to extract, transform and consolidate data. The steps can consist of any combination of connections, tasks, and workflow precedences. Because DTS is built around a series of COM components, many different tools can be utilized

with those components to create a package. A package can be created using the DTS Designer or the DTS Import Export Wizard or programmatically through the DTS Object Model with programming languages such as Microsoft VB or VC++. On top of serving as the instructions and data for COM components to execute a package, the package also contains information used to graphically represent the package. This representation is used by tools such the DTS Designer, allowing for a consistent look and feel.

In addition to being able to be created with many different tools, the COM components that create the package allow it to be saved into a variety of formats, including a Microsoft Visual Basic (.BAS) file, a structured storage file, SQL Server 2000 Meta Data Services, and Microsoft SQL Server 2000. Likewise, the COM components that are the DTS engine allow a variety of clients, including custom applications, command-line applications, and SQL Server applications, to execute a DTS package.

Tasks

Tasks are programmatic steps that are executed within a package to move data, transform data, or execute a command. They are the discrete, atomic individual steps within a package. The combinations of these tasks form the data movement or data transformation solution. The tasks can vary in functionality, ranging from using e-mail to sending out a notification to copying databases or data from one SQL Server to another. Despite the vast range of functionality that they provide, tasks can be organized into just a few equivalence classes:

- Tasks that copy and manage data
- Tasks that transform data
- Tasks that execute jobs
- Custom tasks

Custom user tasks can be written with any COM-supported language such as Microsoft VB or VC++ and then registered with the Register Custom Tasks feature in the DTS Designer. Once registered, the new user-defined DTS task can be used like any other task in the DTS environment. The use of custom tasks allows for greater flexibility in creating data movement solutions. Finally, it should be noted that tasks within the package can be executed sequentially as well as in parallel. This choice depends on the processing needs of the solution.

Transformations

Transformations are such an important feature of DTS that they are often listed as a separate component of a package, but it is important to note that they are performed by tasks. A transformation can be in the form of a single function or a series of functions in a script applied to a column or columns of data. Transfor-mations are used to scrub, change, manipulate, and convert data prior to reaching its final destination. A transformation is the “complete change” of the data from its source to its destination. It is

critical to understand that a transformation does not affect the source data; rather, it affects the destination data only.

Transformations typically change the data type, scale, precision, and size. A transformation is also capable of changing the data from a null value to a newly specified value, or vice versa. In addition, a transformation can check columns meeting specified criteria and apply the transformation to that row or rows only.

Transformation functions can perform other complex operations, too. For example, a transformation function can concatenate the first- and last-name columns to create a new value to insert into a Full Name column on the destination table. In one other example, an ActiveX script could check for a value of Y in the source column and change it to a numeric value of 1 in the destination column. In addition to preprogrammed logic, transformations can also use an external value retrieved at runtime to control the transformation.

Transformations will occur and be used quite frequently in the creation of a data movement solution. DTS encapsulates some of these transformations as functions in most of its supplied tasks. Within these tasks are such functions as the Copy Column, Read File, Write File, and Lowercase String functions. The ActiveX script transformation gives the user additional functionality and control by granting the ability to write scripts to transform, evaluate, and validate the data. If you need additional functionality, custom transformation functions can be created within COM objects and used in the transformation process.

Connections

Connections are used to allow the package to interface with various heterogeneous data sources. They define how to access the source and destination data sources. How you define the data source is important. For instance, defining an Oracle database connection with the OLE DB provider for Oracle will allow different capabilities than defining it with the OLE DB provider for ODBC. The DTS package is usually made up of at least one source and one destination connection.

DTS connections are based on the OLE DB architecture, allowing the use of numerous, readily available OLE DB-compliant providers. In addition to the vast array of OLDE DB drivers, DTS can use ODBC connections through the OLE DB provider for ODBC, and it can access flat files using the OLE DB flat-file provider. DTS has native OLE DB support for both Oracle and SQL Server because they are so widely used. Other providers exist for such data sources as a Microsoft Excel spreadsheets, Microsoft Access, and HTML.

Package Workflow

Any robust application is often a collection of contingency plans. *Package Workflow* is the mechanism that allows a developer to plan for contingency while moving and

transforming data with DTS. For example, a user might not want a certain task to execute if a prior task failed to complete successfully. Instead, the user might want a Send Mail task executed to notify a manager that the task failed to complete successfully. The conditional logic embedded in Workflow can define when to execute steps repeatedly, skip them, or terminate the execution of a package altogether. Package Workflow is the glue that binds the tasks together within a package by allowing programmatic control of flow for the execution of the tasks in a package. Workflow can be defined in two ways:

- Precedence constraints
- ActiveX scripts

Precedence constraints are a control flow-based mechanism that is very similar to event-driven programming. Three constraints each correspond to an event that can be raised relative to a task execution:

- On completion
- On success
- On failure

The On Completion constraint is an unconditional sequential link between two steps in package. This success of the step does not matter; all that matters is that it has been completed. The On Success constraint is used to direct workflow to the destination step when the source step completes successfully. This constraint is useful when you know that one thing is a prerequisite to another. The On Failure constraint, the last type of precedence constraint, is often used as a mechanism for error handling. Commonly, a message might be fired to indicate that a task could not complete because of a failure. Because of their Boolean nature, several tasks' results can be combined to control the execution of another task. For instance, Task C might fire only after Task A has been completed and Task B has succeeded.

Precedence constraints allow for quick, easily understood, powerful control of flow language for DTS. However, sometimes more complex logic might be needed, and in those instances you can use ActiveX scripts. *ActiveX scripts* enhance the control of workflow abilities of packages beyond precedence constraints by including the ability to:

- Restart an entire workflow
- Skip tasks in certain contingencies
- Initiate a threshold for retries of connections and tasks
- Implement loops

The ability to implement loops and skip tasks in certain contingencies allows greater programmatic control of a package. The ability to restart an entire workflow or to retry a task a number of times makes the fault tolerance of the package more robust. In addition to these benefits, ActiveX scripts can be used to initialize and utilize global variables before and during the execution of a step or task.

Workflow is essential to a package and must be carefully thought out. You should plan for all likely contingencies. Poor planning and lack of workflow in a package can cause incorrect results due to the sequencing of the tasks or steps. In addition to proper planning, a healthy amount of testing should be applied to your package to ensure that the package is producing the correct results.

Security in DTS Packages

Due to the far-reaching scope of the resources that may be part of a package and the number of ways a package can be saved and executed, a number of security-related issues are involved with DTS packages. A package can rely on integrated security to interface with data sources. If it does, whoever executes the package must have correct permission to access the package and the resources that the packages will utilize. This process might sound straightforward, but often it is not, because you can execute a package on the server of the client. If you execute it on the client, the Security ID (SID) of the client will be used, but if it were executed on the server using SQLAgent, the SID of SQLAgent would be used. You must isolate the context under which the package will be executed and ensure that the SID will use the correct privileges.

If you do not rely on integrated security, you need to specify the connection information. In this case, you need to make sure that the SID that the connection information will use has the correct authority to access the package and the resources that the package needs. The conundrum here is that other users who can access the package would be able to see the connection information and access the data source on their own. If you save the package as a Visual Basic Script file or to a DTS package file, you can limit access to it using NT Security. If you save it to either SQL Server or Meta Data Services, you can restrict permissions to the data by limiting access to the tables in the msdb database. However, there is another option for packages saved to a DTS package file or to SQL Server: You can encrypt the data in the package by assigning an *owner password* to the package.

In addition to protecting the data inside the package, security issues are involved in limiting who can execute a package. Once more, you can limit access to packages saved as a Visual Basic Script file and DTS package file using NT File Security and packages saved as SQL Server or Meta Data Services by limiting access to the tables in the msdb database. Again, there is another option for packages saved to a DTS package File or to SQL Server: You can restrict execute permission to the package by assigning a *user password* to the package. Again, you must consider the context of who will be executing the package.

DTS Performance Considerations

It goes without saying that performance is always a key to the success of any technology, and this is true of DTS. You can use many techniques to help boost the performance of your DTS system.

One thing that you can do to improve the performance of your DTS solution is to *not* use ActiveX scripts if you do not need them. ActiveX scripts will slow the performance of DTS by adding the overhead of the scripting engine and the processor cycles it takes to iterate through the data. If you implement ActiveX scripts, you should use Microsoft Visual Basic Scripting Edition (VBScript) because it performs better than both JScript and PerlScript. You can increase the performance of your ActiveX scripts by referencing the members of the columns collection by its ordinal position as opposed to its name. Each time you use a column's name, it must be translated into the ordinal position. Although this aids the performance of your script, it should not be taken on lightly, because you sacrifice the readability, flexibility, and maintainability of your code. Whenever you reference the column by its ordinal position, you should place a comment outside the column reference, indicating the name of the field so that the script is easily understood and maintained.

Because all communication is channeled through your data providers, faster data providers provide better performance. In general, a native OLE DB provider is faster than the ODBC provider. In addition, if you reuse a single data connection through many tasks, those tasks cannot be executed in parallel. If you create multiple data connections to the same data source, the tasks that use the various connections can act in parallel, allowing for far better performance than having each task executed serially.

When you are importing data, it is important to note that the Bulk Insert task performs better than using tasks that use BCP or the data pump. This is true because the Bulk Insert task uses the BULK INSERT T-SQL statement. However, you must remember that the BULK INSERT T-SQL statement cannot perform transformations. You can improve the performance of the Transform Data task and the Data Driven Query by improving the performance of the data pump. The performance of the data pump can be optimized using many-to-many column mappings instead of single-to-single column mappings. This will increase speed be decreasing the iterations through the data, but it will sacrifice readability.

In addition to improving the performance of the package by improving the performance of its connections and tasks, you can also improve the performance of the overall package in the way that you save it and execute it. The format in which you save the package can affect the performance of execution; saving a package as a file is the fastest, and saving it to the database is the slowest. In addition, it should be noted that executing a package on the server is often faster than executing a package on the client, because there is less network traffic and, generally, servers have more resources available. You can increase the operating system priority of your package's process by setting it at a higher value in the DTS Package Properties dialog box, as shown in Figure 9.2 . You can access this dialog box in the DTS Designer by selecting Properties on the Package menu.



Figure 9.2: The Package Properties dialog box.

Finally, you can help the performance of DTS applications running on Windows 2000 by setting Turn on Cache for the packages. This can be done in Enterprise Manager as follows:

1. Navigate to the correct server instance in Enterprise Manager, and right-click the Data Transformation Services folder.
2. In the pop-up menu, choose the Properties menu item.
3. In the cache frame, check the Turn on Cache check box.
4. Click the OK button.

Creating and Editing DTS Packages

As mentioned earlier, DTS is really a series of COM-based components that can load and execute a package. Similarly, those DTS components can also be used to create packages. Because the COM objects do not have a presentation layer embedded in them, other tools use them to create DTS packages, allowing for multiple user interfaces. Although you can access the DTS objects directly to create a package with a programming tool, typically two tools are used to create a DTS package:

- The DTS Import/Export Wizard
- The DTS Designer

The DTS Designer and the DTS Import/Export Wizard both have the ability to create packages. Often, the DTS Import/Export Wizard is the fastest way to create a data movement solution, but the DTS Designer gives you more control and power. Although both tools can create packages, only the DTS Designer can edit existing DTS packages.

Creating a Simple Package

1. Navigate to the correct server instance in Enterprise Manager, and right-click the Data Transformation Services folder.
2. In the pop-up menu, choose the New Package menu item.
3. In the DTS Designer, click the SQL Server symbol in the connection toolbar.
4. Enter **Northwind** in the New Connection text box. Choose Microsoft OLE DB Provider for SQL Server as the data source. Choose a valid server and the valid connection information in the SQL Server frame. Select Northwind in the Database list.
5. Click the OK button.
6. Enter **Pubs** in the New Connection text box. Choose Microsoft OLE DB Provider for SQL Server as the data source. Choose a valid server and the valid connection information in the SQL Server frame. Select Pubs in the Database list.
7. Click the OK button.
8. Click Transform Data Task.
9. Click the Northwind connection when prompted for the source and the Pubs connection when prompted for the destination.
10. Right-click the arrow that connects the Northwind connection to the Pubs connection, and select the Properties menu item.
11. In the Transform Data Task Properties dialog box, select Item as the Table/View list. Click the Destination tab.
12. Click the Create button to the left of the Table name list.
13. Click the OK button.

Often, when you create a package, a great deal of the logic that the package needs might already exist within another package. In early versions of DTS, the only way that DTS could reuse packages was via the ActiveX script or Execute Process tasks. The ActiveX Script task allowed you to invoke the package just as you would in a custom application. It allowed full use of the object model and provided a very robust interface to the existing package. In addition, the Execute Process task allowed you to execute a task using the DTS Run application to run the package. This is far faster to develop, but as a consequence, you had less power over the package.

Now you can use the Execute Package task. This task is as fast to develop as the Execute Process task, but it is more naturally incorporated into the workflow of your current package, since it was designed to execute a package. You can still use the ActiveX script or Execute Process task, but the Execute Package task is easier to deal with, allows the child transaction to participate in the parent transaction, and allows the child to access the global variables of the parent.

Reusing existing logic can cut down on development time and, in some ways, can make maintenance easier because you do not have to duplicate changes. However, if you're not responsible for the package that you are trying to recycle, it can also complicate maintenance. The person who maintains the package might need to change it in the future to satisfy his or her requirements, altering the output that you're expecting. No matter who maintains the package, though, you need to clearly document the dependency if you reuse a package.

DTS Import/Export Wizard

Part of Microsoft's development philosophy is to make the common case easy and fast. Typically, the company accomplishes this goal with the use of wizards, and DTS is no different. DTS includes the DTS Import/Export Wizard, which collects all the needed information and creates a package. Although the wizard can create a package, it is not a vehicle via which to edit a package.

The wizard steps the user through a series of screens asking for source and destination information. The wizard also allows the user to make necessary transformations that will occur to the data during the import or export. The wizard allows the user to save, schedule, or run the package. It also allows the user to import and transform heterogeneous data.

Copying a Table Between Databases

1. Click the Wizard icon on the toolbar or select the Wizards menu option from the Tools menu to display the Select Wizard dialog box.

2. Select the DTS Export Wizard option from the Database Transformation Services group, and click the OK button to continue.
3. Read the splash screen of the DTS Import/Export Wizard, and click the Next button.
4. Specify the connection information for the source data source by choosing a data-specific driver from the Data Source list and filling out the security and location information specific to the driver, as shown in Figure 9.3 .

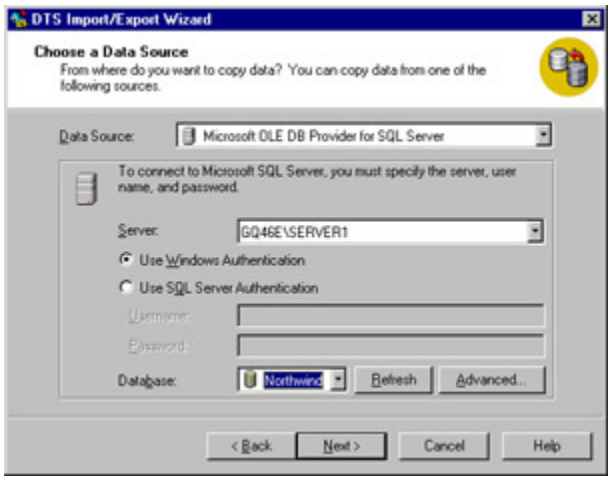


Figure 9.3: DTS Export Wizard data source selection.

5. Specify the connection information for the destination data source by choosing a data-specific driver from the Data Source list and filling out the security and location information specific to the driver.
6. Choose the type of data transfer you want to execute by selecting “Copy table(s) and view(s) from the source database,” “Use a query to specify the data to transfer,” or “Copy objects and data between SQL Server databases.”
7. If you select “Copy table(s) and view(s) from the source database,” you will need to select the tables and views from a list.

DTS Designer

The DTS Designer can also create a package, but more important, it is your primary avenue to edit packages. It is a Microsoft Management Console (MMC) snap-in that is integrated into Enterprise Manager, as shown in Figure 9.4 . DTS Designer is a graphically oriented design tool that allows you to edit quickly and with ease. All the available tasks and connections are laid out on the Tasks and Connections toolbars, as shown on the right-hand side in Figure 9.4 . To add a task or connection to the package, a developer need only drag and drop the symbol on the icon, then fill out the information for which the system prompts.

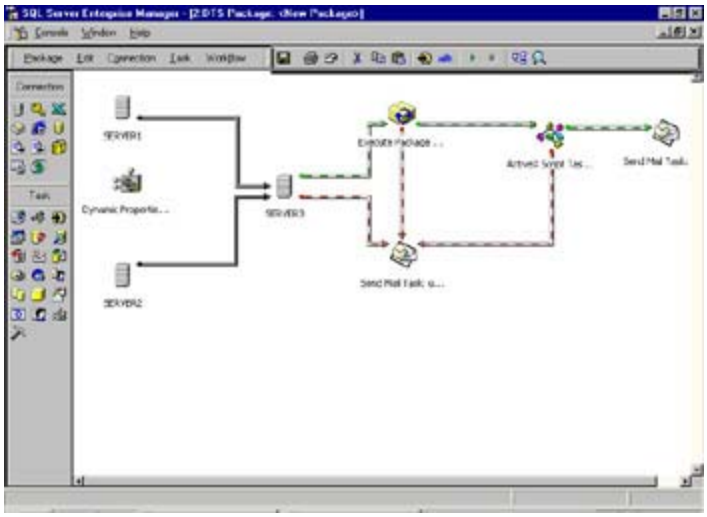


Figure 9.4: The DTS Designer.

Creating a Data Connection

1. In the DTS Designer MMC, drag the connection you want to create from the Connection toolbar onto the DTS Designer design sheet.
2. In the Connection Properties dialog box, choose whether you want a new or existing connection. If you choose New Connection, you need to enter a name for the connection in the New Connection text box. If you choose Existing Connection, you must select a connection from the Existing Connection list.
3. Choose an OLE DB provider from the Data Source list.
4. Fill in the security and connection information that is specific to the data source you have chosen.
5. Click the OK button.

Creating a Task

1. In the DTS Designer MMC, drag the connection you want to create from the Task toolbar onto the DTS Designer design sheet.
2. In the Task Properties dialog box that pops up, fill out the information that is required by each task type.
3. Click the OK button.

Types of Tasks

As stated earlier, DTS tasks can be fundamentally organized into several equivalence classes:

- Tasks that copy and manage data
- Tasks that transform data
- Tasks that execute jobs
- Custom tasks

The first class of tasks is the one that copies and manages data, and it is listed in Table 9.1 . These tasks are used to move and manage data from the data source(s) to the destination data source(s). These are the bread-and-butter operations in DTS and are very commonly used. In addition to moving and managing the data inside data structures, this class includes tasks that can move and manage the data that define database objects. You cannot move only the data in a table; you also move the table structure itself. In addition to moving a table, you can move and manage views, logins, jobs, databases, and stored procedures.

Table 9.1: Tasks That Copy and Manage Data

Task	**Description**	**New**	**Transactional Support**
Bulk Insert	Works like the T-SQL BULK INSERT command. Copies large amounts of data from a text file into a SQL Server table or view. No transformations occur.	—	Connection dependent
Copy SQL Server Objects	Copies objects from one SQL Server to another SQL Server. This includes copying objects from one instance to another instance of SQL Server on the same server. The objects include table and data, stored procedures, views, rules, defaults, user-defined functions, and user-defined data types.	—	No
Execute SQL	Executes SQL statements. Query data, output parameters, and return values can be saved to global variables.	—	Connection dependent
Transfer Database	Copies or moves the entire database from a SQL Server 2000 or SQL Server 7.0 server to a SQL Server 2000 server. This task cannot copy a database from a SQL Server 2000 server to a SQL Server 7.0 server.	Yes	Yes
Transfer Error Messages	Copies user-defined error messages. These are messages that have been added using the sp_addmessage stored procedure. Error messages can be copied from SQL Server 7.0 and SQL Server 2000 to SQL Server 2000 only.	—	Yes
Transfer Jobs	Copies jobs from one SQL Server instance to another. Jobs can be copied from SQL Server 7.0 and SQL Server 2000 to SQL Server 2000 only.	Yes	Yes

Table 9.1: Tasks That Copy and Manage Data

Task	Description	New	Transactional Support
Transfer Logins	Copies logins from one SQL Server instance to another. Logins can be copied from SQL Server 7.0 and SQL Server 2000 to SQL Server 2000 only.	—	Yes
Transfer Master Stored Procedures	Copies master stored procedures from one SQL Server instance to another. Master stored procedures can be copied from SQL Server 7.0 and SQL Server 2000 to SQL Server 2000 only.	Yes	Yes

The next task grouping, displayed in Table 9.2 , to considered the one that transforms data. These tasks manipulate and change the data while they are being moved or copied to a new location. Instead of trying to put a square peg in a round hole, these tasks allow you to transform your “square data” into “round data” so that the data in the data structures are described in a manner that can be understood. For instance, the source data could track priority with numbers from 1–255, but the destination data source allows only values up to 16 in its priority field. You can use transformation to massage the data as you move them to fit in the structures.

Warning If the Transfer Database task is used to move a database, it detaches the database from the source SQL Server. However, you can’t copy or move the database back to a SQL Server 7.0 Server. Fortunately, the task does not delete the data and log files. You must use sp_attach_db to reattach the database on SQL Server 7.0 or restore from backup.

Table 9.2: Tasks That Transform Data

Task	Description	New	Transactional Support
Data Driven Query	Executes one of the following: stored procedure, INSERT, UPDATE, or DELETE against one row of the data set at a time. The operation is based on criteria returned from an ActiveX script at runtime. Allows the choice of T-SQL statements to be dynamic for each row.	—	Connection dependent
Transform Data	Copies and transforms data between a source and destination. The transformations are applied at the column level.	—	Connection dependent
Parallel Data Pump	Works like the Transform Data task and the Data-Driven Query task. Can be	Yes	Yes

Table 9.2: Tasks That Transform Data			
Task	**Description**	**New**	**Transactional Support**
	used only programmatically and supports the use of chaptered rowsets defined in OLE DB 2.5 and later.		

The next category of tasks is the one that executes jobs; it is shown in Table 9.3 . These tasks often provide interfaces to external systems such as mail or message queues. As a consequence, most of the tasks will not participate in a transaction. For instance, if you send an e-mail saying that you have completed the process, but later the transaction aborts, the e-mail will not be rolled back. Among these tasks are the Microsoft ActiveX task, shown in Figure 9.5 , which provides customizable scripts in languages such as Microsoft VBScript and Microsoft JScript. The use of scripts can be a very flexible and powerful advantage, but they carry a drawback: The script is applied to each row during execution and could significantly affect the performance of a package.

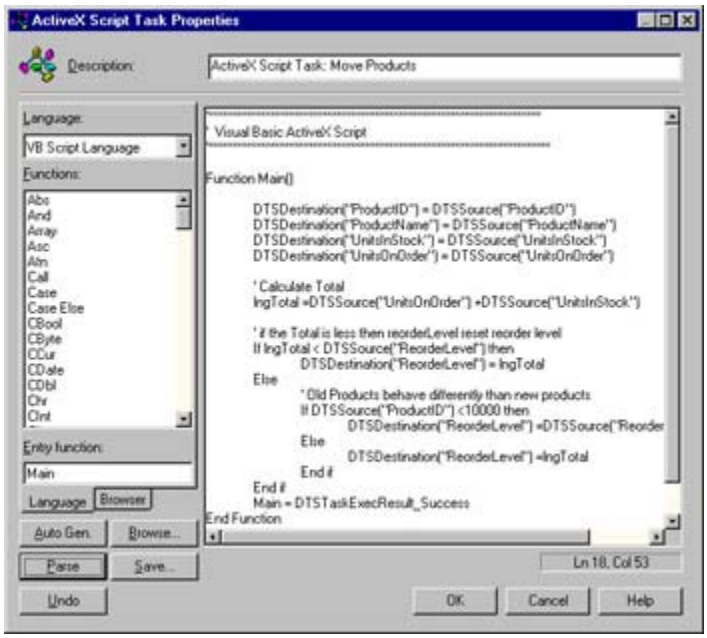


Figure 9.5: The ActiveX Scripts task.

Finally, there are the Custom tasks. These are user-defined tasks that can do almost anything. They are written in programming languages such as Visual Basic. These tasks are added via the Register Custom task in the DTS Designer or by using the DTS Object Model.

Table 9.3: Tasks That Execute Jobs			
Task	**Description**	**New**	**Transactional Support**

Table 9.3: Tasks That Execute Jobs

Task	Description	New	Transactional Support
ActiveX Script	Used to perform operations of which the other tasks are not capable. This is probably the most powerful of all the DTS-supplied tasks. The ActiveX Script Task Properties dialog box is used to enter valid DTS script commands, as shown in Figure 9.5 .	—	No
Analysis Services	Used to process objects within SQL Server 2000 Analysis Services.	Yes	Yes
Data Mining	Used to create a prediction query and output table. The query and output table is generated from a data-mining object within SQLServer2000AnalysisServices.	Yes	Yes
Dynamic Properties	Used to get values outside DTS packages. The values are assigned to specific properties of the package.	Yes	No
Execute Package	Used to execute DTS packages from within another DTS package.	Yes	Connection dependent
Execute Process	Used to run application executables or batch files from within a DTS package.	Yes	No
FTP	Used to download files from remote FTP sites.	—	No
Message Queue	Used to send and receive messages from a Microsoft Message Queue.	Yes	Connection dependent
Send Mail	Used to send e-mail messages.	Yes	No

Saving DTS Packages

When you save a DTS package, you preserve all DTS objects with the properties and relationships and the graphical layout of the DTS Designer workspace. You can save a DTS package to a number of formats:

- SQL Server
- Meta Data Services
- DTS Package File
- Visual Basic Script File

Saving your package to a SQL Server will store the information in the msdb database. One of the main advantages of saving it as a package to the SQL Server is that the centralized storage allows centralized access. In order to keep sensitive information confidential, you can encrypt the package by assigning it an owner password and limit the execution to certain people by assigning a user password. Saving the package to Meta Data Services also allows for centralized storage, but it does not offer the same security features. Saving it as DTS package file does allow the same security features as saving it to a SQL Server, but it is saved to a single file instead of to a server. Saving it to a Visual Basic script file allows for easy editing but no additional security features.

If you saved the package to SQL Server 2000 or SQL Server 2000 Meta Data Services, DTS will track its past versions. These packages can be edited and resaved as needed. However, you can delete past versions of a DTS package only from SQL Server 2000.

Saving to a SQL Server

1. Click the Save menu item on the Package menu in the DTS Designer MMC.
2. In the Save DTS Package dialog box, enter a valid name for the DTS package in the Package name text box.
3. If you want to encrypt server password and username information within the package, enter an owner password in the Owner password text box.
4. If you want to limit access to the execution of the file, set a user password in the User password text box. You must set an owner password if you want to enter a user password value.
5. Choose a SQL Server from the Location combo box, as shown in Figure 9.6 . If the SQL Server instance that you want to save to is not in the list, you can type it into the combo box.

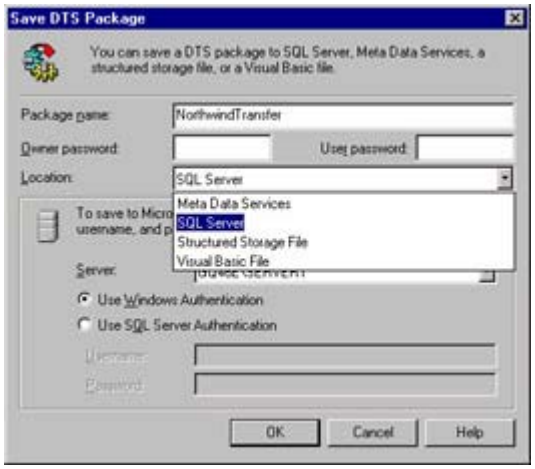


Figure 9.6: The Save DTS Package dialog box.

6. Select the type of authentication information needed to access the SQL Server.
7. Click the OK button.

Saving to Meta Data Services

1. Click the Save menu item on the Package menu in the DTS Designer MMC.
2. In the Save DTS Package dialog box, enter a valid name for the DTS package in the Package name text box.
3. Choose a Meta Data Service from the Location combo box, as shown in Figure 9.6 .
4. Select a SQL Server from the server list; if the SQL Server instance that you want to save to is not in the list, you can type it into the combo box.
5. Click the Scanning button next to the server.
6. In the Scanning Options dialog box, specify how the package should be related to the catalog's metadata by choosing the catalogs that should be scanned.
7. Click the OK button.
8. Select the type of authentication information needed to access the SQL server.
9. Click the OK button.

Saving to a DTS Package File

1. Click the Save menu item on the Package menu in the DTS Designer MMC.
2. In the Save DTS Package dialog box, enter a valid name for the DTS package in the Package name text box.
3. If you want to encrypt server password and username information within the package, enter an owner password in the Owner password text box.
4. If you want to limit access to the execution of the file, set a user password in the User password text box. You must set an owner password if you want to enter a user password value.
5. Choose a structured storage file from the Location combo box, refer back to Figure 9.6 .
6. Choose a valid filename in the Filename text box.
7. Click the OK button.

Saving to a Visual Basic Script File

1. Click the Save menu item on the Package menu in the DTS Designer MMC.
2. In the Save DTS Package dialog box, enter a valid name for the DTS package in the Package name text box.
3. Choose a Visual Basic file from the Location combo box, refer back to Figure 9.6 .
4. Choose a valid filename in the Filename text box.
5. Click the OK button.

Executing DTS Packages

Once all the tasks have been created and the package has been saved, it can be executed immediately in the DTS Designer and SQL Server Enterprise Manager. The package can be run at the completion of the Export and Import Wizard, or it can be executed on schedule using SQL Server Agent. In addition, a DTS package can be executed from the command prompt with the utility DTS Run (dtsrun.exe) or with the DTS Run Utility (dtsrunui.exe). Finally, it can be executed programmatically using the Execute method of the Package object.

One of the important factors to consider when you choose the manner in which you want to execute the package is the context in which the package will run. Executing the package from the DTS Designer Interface and the Export and Import Wizard causes the package to execute in the memory space of SQL Enterprise Manager. Executing a package from dtsrun.exe or dtsrunui.exe runs it in the local memory space of dtsrun.exe or dtsrunui.exe. If you programmatically invoke a package through the Package object, it runs in the space of the application. All these methods run the package on the client, creating additional network traffic and potentially causing problems because the client might not be configured in the same way as the server. For instance, the client application might not have the correct privileges to access all the resources. In addition, the client needs to refer to the resources that the package needs in the same manner as the package; if a drive is referred to by drive letter, for example, the client needs to have the same drive letter assigned to that drive.

Executing the package through SQL Server Agent executes the package in the memory space of the SQL Server Agent on the server. This method should perform better because it reduces the amount of network communication and alleviates security and reference problems. As mentioned, you can execute a package in a number of different ways, refer to the following sections.

Executing a Package in the DTS Designer Interface

1. Open the package in the DTS Designer.
2. Click the Execute menu item on the Package menu, or click the Execute button.
3. Monitor the progress of your package in the Executing Package dialog box, and click the OK button on the Package Execution Results dialog box.
4. Click the Done button.

Executing a Package in SQL Enterprise Manager

1. Right-click the package in the appropriate folder (Local Package, Meta Data Services Packages, or Meta Data Package) on the correct server instance in the Data Transformation folder.
2. Choose the Execute Package menu item.
3. Monitor the progress of your package in the Executing Package dialog box, and click the OK button on the Package Execution Results dialog box.

4. Click the Done button.

Executing a Package Using the dtsrun.exe Utility

1. Open a command window.
2. Type in **dtsrun** with the package name and the proper arguments, then press Enter. /N is the argument that used to specify to the package name. If you need to connect to the SQL Server, you can use /S to specify the server name, the /U argument to specify the username, and /P to specify the password. If the package is stored in a structured storage file, you need to specify the file with the /F argument. A complete list of arguments that can be used with dtsrun.exe is contained in Table 9.4 .

Table 9.4: Arguments Used with dtsrun.exe

Argument	Parameter	Description
/E	None	Specifies that a trusted connection will be used. A username and password do not need to be provided.
/F	Filename	The UNC filename that specifies that location and name of the package.
/G	Package GUID	The GUID that identifies the package.
/M	Package Password	A password needed to execute the package if the package itself has been secured.
/N	Package Name	The name of the DTS package.
/P	Password	The password for gaining access to the SQL Server.
/R	Repository Database Name	The name of the repository on which the DTS package resides.
/S	Server Name	The server name and instance.
/U	User Name	The login name needed to gain access to the SQL Server.
/V	Package Version GUID	The version ID assigned to the package when it was first saved or executed.
~	/S, /U, /P, /F, /R, /N, /M, /G, and /V arguments with their values	Indicates that the following arguments will be encrypted.
/!C	None	Places the command executed from the command line on the Clipboard so that it can be pasted elsewhere.
/!D	None	Deletes the package from the SQL Server. It does not execute the package.

Table 9.4: Arguments Used with dtsrun.exe		
Argument	**Parameter**	**Description**
/!X	None	Overwrites a DTS package file with a DTS package from SQL Server. It does not execute the package.
/!Y	None	Displays the encrypted command that was executed.
/?	None	Displays the available switches.

Executing a Package Using the dtsrunui.exe Utility

1. Open a command window.
2. Type **dtsrunui** and press Enter.
3. When the DTS Run application comes up, as shown in Figure 9.7 , select the location of the package.



Figure 9.7: The dtsrunui.exe user interface.

4. If you selected either SQL Server or Meta Data Services as the location, you need to enter the server and authentication information to connect to the server. If you chose Structured Storage File, you need to enter the filename in the Filename text box.
5. Enter the package name in the Package Name text box. If you do not know the name, you can click the "…" button to the right of the Package Name text box to choose one from the Select Package dialog box.
6. Click the Run button.
7. Monitor the progress of your package in the Executing Package dialog box, and click the OK button on the Package Execution Results dialog box.
8. Click the Done button.

Executing a Package Programmatically in Visual Basic

1. Open Visual Basic.
2. Create a reference to the Microsoft DTS Package Object Library (dtspkg.dll).
3. Instantiate the variables, including one as DTS.Package2.
4. Write code that fills out the needed properties that specify the package, including the FileName, Name, ServerName, ServerPassword, and ServerUserName.
5. Write a line of code after the properties that execute the package with the Execute method.

The Bulk Copy Program

The *Bulk Copy Program (BCP)* is a command-line utility designed to allow SQL Server to interchange data with a file. Although data can be exported or imported, the interchange can happen only in one direction at a time. The data can either be exported from SQL Server to a data file or imported to SQL Server from the file.

The BCP utility is installed in the \MSSQL\Binn, the \80\Tools\Binn, and the \MSSQL$ *instance_name* \Binn directories for each named instance of SQL Server 2000. Any copy of these utilities can be used to connect to any instance of SQL Server, because you choose the instance of SQL Server to connect to by passing server parameters with a command-line switch.

Due to the command-line nature of the BCP application, the user does not require knowledge of T-SQL. However, to use BCP effectively does require the operator to have an understanding of the schema of the database tables with which the BCP application will interact. The database schema is defined in the system tables of SQL Server, and it is inflexible as both a destination and a source. However, the files with which the BCP interacts can be in numerous user-defined formats. You can define the file format interactively, with a switch, or using a format file. All these methods describe how the fields and rows are organized by providing the following data:

- Field delimiter(s)
- Field storage type(s)
- Field prefix length(s)
- Field length(s)
- Row delimiter

Although a format file allows you to use many different formats of data, the format file itself is an inflexible tab-delimited file. Here is what a format file could look like for the Shippers table in the Northwind database:

```
8.0
3
1  SQLINT   0  4   "\t" 1  ShipperID      ""
2  SQLCHAR  2  80  "\t" 2  CompanyName    "Latin1_General_CI_AS"
3  SQLCHAR  2  48  "\t" 3  Phone       SQ "Latin1_General_CI_AS"
```

On the very first line, you will see 8.0. That is the version number of the BCP application. Immediately below it is the number 3, which is the number of fields that will be defined in the subsequent rows. Each subsequent row represents a field and will contain the position of the field in the file, the SQL data type of the field, the prefix length of the field, the maximum length of the field, the delimiter, the ordinal position of the field in the table, the name of the column in the table, and collation used to store character data.

If you examine the row under the 3 (the third row in the file), you can see how the values correspond to the parameters:

```
1   SQLINT    0   4    "\t" 1   ShipperID       ""
```

The 1 is the position that data will take in the file. If you export data with the format file, this will be the first field. If you import data, the data are coming from this field.

The next three fields describe those data in the database. The first one, SQLINT, is the SQL data type of the field. This is how the data are represented in SQL Server; you can see a complete list of the values and their counterparts in the format file in Table 9.5 .

The next two fields describe the length, in bytes, needed to represent the data. To the right of SQLINT is the prefix length—in this case, 0. The prefix length is a value from 0–4 that describes the number of bytes needed to depict this field in addition to the data itself. This is the header information for the field that contains the length of the data and whether it is null. If a field is nullable, the prefix length is at least 1, because you need a byte to indicate whether a value exists or not. For strings and other complex data types, the space allocated by the prefix length is used to store the actual length of the field. For instance, a column's data type might be varchar(80) field, but a value uses only 40 characters. The actual length of the data is stored in the prefix length. The next value you can see is 4, which represents the length of the data type in bytes.

Table 9.5: Data Type Conversions for Format Files

SQL Server Type	**Host File Data Type**
Strings	
Char	SQLCHAR
Nchar	SQLNCHAR
Strings	
Nvarchar	SQLNCHAR
Varchar	SQLCHAR
Text	SQLCHAR
Ntext	SQLNCHAR
Binary	
Binary	SQLBINARY
Varbinary	SQLBINARY
Image	SQLBINARY
Datetime	SQLDATETIME
DateTime	
Smalldatetime	SQLDATETIM4
Timestamp	SQLBINARY

Table 9.5: Data Type Conversions for Format Files

SQL Server Type	Host File Data Type
Numbers	
Decimal	SQLDECIMAL
Numeric	SQLNUMERIC
Float	SQLFLT8
Real	SQLFLT4
Int	SQLINT
Bigint	SQLBIGINT
Smallint	SQLSMALLINT
Tinyint	SQLTINYINT
Money	
Money	SQLMONEY
Smallmoney	SQLMONEY4
Miscellaneous	
Bit	SQLBIT
Uniqueidentifier	SQLUNIQUEID
sql_variant	SQLVARIANT

After the length of the data type, the next value is \t, which is the delimiter; \t represents a tab, so this indicates that the end of this field is signaled by a tab. A list of common delimiters is shown in Table 9.6 . If we had wanted to indicate the end of this field by a carriage return, we would have used \r.

The next two fields specify the location of the data within the table. The first field is the ordinal position of the field. In this case, it is the first position, because the value is 1. Next to it is the name of the field, ShipperID. The final field is blank because it is the collation, and integers do not have collation values. The row below (fourth from the top) is a string and has a collation of Latin1_General_CI_AS. The collation used is Latin1 General dictionary with sorting rules that are case insensitive and accent sensitive.

Table 9.6: Common Delimiters

Symbol	Description
\t	Tab
,	Comma
\n	New line
\r	Carriage return
\\	Backslash

Table 9.6: Common Delimiters	
Symbol	**Description**
\0	Null terminator

Using BCP

If we wanted to execute a very simple bulk copy out of the Products table in the Northwind database, we could issue this statement in a command window:

```
bcp "Northwind..Products" out "Prod.tbl" -c -S"GQ46E\Server1" -P""
```

This command would copy all the rows of the Products table in the Northwind database on the server instance GQ46E\Server1 into a file named Prod.tbl. The –c switch tells the BulkCopy application to treat all data as character data type, with no prefixes. It also dictates that the fields will be tab delimited and the rows will be new-line delimited. The –S switch instructs the BCP application that GQ46E\Server1 is the server instance to use, and the –P switch tells it that we want to use the password of "". If we omit the –c and –P switches like this, the BCP application will prompt us for the format and password:

```
bcp "Northwind..Products" out "Prod.tbl" -S"GQ46E\Server1"
```

The BCP application will prompt for the format and password. After you have provided it with a valid password, it will iterate through all the columns in the table, asking you for the file storage type, prefix length, and field terminator, like this:

```
Enter the file storage type of field ProductID [int]:
Enter prefix-length of field ProductID [0]:
Enter field terminator [none]: \t

Enter the file storage type of field ProductName [nvarchar]:
Enter prefix-length of field ProductName [1]:
Enter field terminator [none]: \t

Enter the file storage type of field SupplierID [int-null]:
Enter prefix-length of field SupplierID [1]:
Enter field terminator [none]: \t

Enter the file storage type of field CategoryID [int-null]:
Enter prefix-length of field CategoryID [1]:
Enter field terminator [none]: \t

Enter the file storage type of field QuantityPerUnit [nvarchar]:
Enter prefix-length of field QuantityPerUnit [1]:
Enter field terminator [none]: \t

Enter the file storage type of field UnitPrice [money-null]:
Enter prefix-length of field UnitPrice [1]:
Enter field terminator [none]: \t

Enter the file storage type of field UnitsInStock [int-null]:
```

```
Enter prefix-length of field UnitsInStock [1]:
Enter field terminator [none]: \t

Enter the file storage type of field UnitsOnOrder [int-null]:
Enter prefix-length of field UnitsOnOrder [1]:
Enter field terminator [none]: \t

Enter the file storage type of field ReorderLevel [int-null]:
Enter prefix-length of field ReorderLevel [1]:
Enter field terminator [none]: \t




Enter the file storage type of field Discontinued [bit]:
Enter prefix-length of field Discontinued [0]:
Enter field terminator [none]: \r\n
```

You will notice that often, we did not enter a value. If you use the Enter key to continue, the default value (which is specified by the brackets at the end of the question) will be the value that is used. These suggestions are based on the BCP application interrogating the table involved to derive what it believes are optimal values. After BCP has finished iterating through all the columns, it will allow you to save you answers to a format file. A *format file* is one that can be used in future executions so that you do not have to answer these questions again. If you answer yes (Y), you will be prompted for a file name, as shown here:

```
Do you want to save this format information in a file? [Y/n] Y
Host filename [bcp.fmt]: prod.fmt
```

Once you enter the filename, the bulk copy out of the database will commence, and you will have two output files: Prod.tbl and Prod.fmt. The advantage of saving your output to a format file is that you can run the export with the answers that you specified using the format file via the –f switch, like this:

```
bcp "Northwind..Products" out "Prod.tbl" -S"GQ46E\Server1" -T -
f"prod.fmt"
```

Note Command-line application switches are case sensitive. The –c switch dictates information about the format of data type, prefixes, and delimiters of a file, whereas the –C switch dictates what code page to use. It is very important for you to pay attention to the case of the switches that you use with the BCP command-line application.

In addition to copying information out of the database with the BulkCopy application, you can copy information back into the database. Unlike copying information out of the database, when you copy into the database, the destination—in this case, a table—must already exist; if a file did not exist, BCP would create it, but it is not allowed to create tables. In addition, in order to copy data into the database, you must have permission to insert records into the table.

So that we do not interfere with the existing Products table in the Northwind database, let's create a table with a similar structure, called Products2. You can create Products2 with this script:

```
if exists (
            select *
            from dbo.sysobjects




            where id = object_id(N'[dbo].[Products2]'
            )
  and OBJECTPROPERTY(id, N'IsUserTable') = 1)
  drop table [dbo].[Products2]
GO


CREATE TABLE [dbo].[Products2] (
   [ProductID] [int] IDENTITY (1, 1) NOT NULL ,
   [ProductName] [nvarchar] (40) COLLATE SQL_Latin1_General_CP1_CI_AS
NOT NULL ,
   [SupplierID] [int] NULL ,
   [CategoryID] [int] NULL ,
   [QuantityPerUnit] [nvarchar] (20) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL ,
   [UnitPrice] [money] NULL ,
   [UnitsInStock] [smallint] NULL ,
   [UnitsOnOrder] [smallint] NULL ,
   [ReorderLevel] [smallint] NULL ,
   [Discontinued] [bit] NOT NULL
) ON [PRIMARY]
GO


ALTER TABLE [dbo].[Products2] WITH NOCHECK ADD
   CONSTRAINT [PK_Products2] PRIMARY KEY CLUSTERED
    (
        [ProductID]
     )  ON [PRIMARY]
GO
```

Now that we have this new table, we can import the data into Products2 that we exported from the Products table, like this:

```
bcp "Northwind..Products2" in "Prod.tbl" -c -S"GQ46E\Server1" -P""
```

In order to copy data into the database, we replaced the *out* keyword between the database and file with the *in* keyword. If you look at the results, you will see that all 77 rows are in the database, and everything appears as it should be. Now delete all the rows with this Transact-SQL statement:

```
DELETE FROM Products2
```

Now run the BCP statement again. When you look at the data, you will notice that the ProductID fields start at 78 and go through 144; these are not the values in your file. This is a result of the ProductID value being generated from an IDENTITY; even though you have specified a value, the identity is overriding that value and generating a value for this column. As you can imagine, this can cause quite a few problems with referential integrity while you are moving data around. There is a solution to this problem, however: the –E switch. The –E switch instructs SQL Server not to use the values generated by IDENTITY and instead to use the values in the file. You can use the –E switch in this statement to import the values with the correct ProductID:

```
bcp "Northwind..Products2" in "Prod.tbl" -c -S"GQ46E\Server1" -P"" –E
```

In addition to using a format file to help you export data, you can also use it to help you import data, like this:

```
bcp "Northwind..Products" in "Products.tbl" -S"GQ46E\Server1" -P"" -
f"prod.fmt"
```

SQL-DMO BulkCopy

The BCP application provides a great deal of functionality, but it is hindered, in some ways, by being a command-line application. If you want to be able to automate many repetitive BCP tasks, the BulkCopy and BulkCopy2 SQL-DMO objects are much more useful to you. The BulkCopy and BulkCopy2 objects encapsulate all the parameters needed for the ImportData and ExportData methods of the Table SQL-DMO object and the ExportData method of the View SQL-DMO object. In addition to providing information to the Table and View SQL-DMO, BulkCopy and BulkCopy2 can also cause the bulk copy to abort through the Abort method and can return status information through the BatchImported and RowsCopied Events.

The BulkCopy object existed in previous iterations of SQL Server, but BulkCopy2 was added with SQL Server 2000. Through inheritance, BulkCopy2 has all the methods, events, and properties of the original BulkCopy object. It has added a new property, Tablock, that allows you to issue a table-level lock while you are importing data, thus allowing for better performance.

Using the BulkCopy Object

In order to use the BulkCopy object with Visual Basic, you must access the Microsoft SQLDMO Object Library. You can access it using late binding with the CreateObject syntax, like this:

```
Dim objSQLServer2 as Object
Set objSQLServer2 = CreateObject("SQLDMO.SQLServer2")
```

Or you can early bind to it by creating a reference to sqldmo.dll. It will appear as the Microsoft SQLDMO Object Library in the References list. *Early binding* provides significantly better performance and simplifies new development with the availability of Intellisense. Once you have made a reference to the library, you would early bind to the SQLDMO.SQLServer2, like this:

```
Dim objSQLServer2 As SQLDMO.SQLServer2
Set objSQLServer2 = New SQLDMO.SQLServer2
```

Events and methods are associated with the BulkCopy objects, but for the most part they are used as containers for the properties that define BulkCopy operation. Since the BulkCopy objects are used with the Table and View objects, the BulkCopy objects do not define or have knowledge about the data source's name, structure, or location of the information in the SQL Server. However, they do have full control over the structure, name, and location of the files in the BCP process through the DataFilePath property. A valid value must be set with the DataFilePath for the BulkCopy and BulkCopy2 objects to succeed. We could implement the BulkCopy object as follows:

```
Dim objBulkCopy       As SQLDMO.BulkCopy
Dim objSQLServer2     As SQLDMO.SQLServer2
```

```
Dim objProducts2      As SQLDMO.Table
Dim lngNumberOfRows   As Long

    Set objSQLServer2 = New SQLDMO.SQLServer2
        objSQLServer2.Connect "GQ46E\SERVER1", "sa", ""

        Set objBulkCopy = New SQLDMO.BulkCopy
            With objBulkCopy
                 .DataFilePath = "C:\prod.tbl"
                 .DataFileType = SQLDMODataFile_CommaDelimitedChar
            End With
            Set objProducts2 = _
objSQLServer2.Databases("Northwind").Tables("Products2")
            objProducts2.ExportData objBulkCopy
            Set objProducts2 = Nothing
    Set objBulkCopy = Nothing
  Set objSQLServer2 = Nothing
```

The first thing that we did was to connect to the SQL Server instance by giving the Connect method the name of the server instance (GQ46E\SERVER1), login (sa), and password (“”). Next you see the creation of the BulkCopy object and the date file set with the DataFilePath and the format of the data set with the DataFileType. The DataFileType accepts an enumeration listed in Table 9.7 , a table of DataFileType property enumeration values, to control the definition of the file. With it you can set a predefined format, a format defined by a format file, or a format defined by other properties of BulkCopy. If you set the DataFilePath to SQLDMODataFile_SpecialDelimitedChar, you need to set the ColumnDelimiter and RowDelimiter properties. If you set the DataFilePath to SQLDMODataFile_ NativeFormat, you need to set the Use6xCompatible property to establish whether you want to use the native data format before 7.0 or after 7.0. Once you have set all the values that you choose, you pass the BulkCopy object to a Table or View object. In this case, we passed it to a Table object that represented the Products2 table in the Northwind database.

Table 9.7: Data File Type Property Enumeration Values

Enum	Enum Name	Description Value
1	SQLDMODataFile_CommaDelimitedChar	A comma delimits each column. Each row is delimited by a carriage return or line feed.
1	SQLDMODataFile_Default	It is the same as the SQLDMODataFile_CommaDelimitedChar. A comma delimits each column, and each row is delimited by a carriage return or line feed.
2	SQLDMODataFile_TabDelimitedChar	A tab delimits each column, and each row is delimited by a carriage return or line feed.
3	SQLDMODataFile_SpecialDelimitedChar	A user-defined string defined by the

Table 9.7: Data File Type Property Enumeration Values

Enum	Enum Name	Description Value
		ColumnDelimiter property delimits each column, and each row is delimited by user-defined string defined by the RowDelimiter property.
4	SQLDMODataFile_NativeFormat	File data will be defined in the SQL Server bulk copy native format.
5	SQLDMODataFile_UseFormatFile	The format file identified in the FormatFilePath property of the BulkCopy object will be used to describe the data.

As indicated before, you could have used a format file that we used with the BCP application. In order to utilize a format file, you need to set the DataFileType to SQLDMODataFile_UseFormatFile and then set the FormatFilePath to the location of the format file. You could reuse the prod.fmt format file that we created with the BCP application, like this:

```
With objBulkCopy
     .FormatFilePath = "C:\prod.fmt"
     .DataFilePath = "C:\prod.tbl"
     .DataFileType = SQLDMODataFile_UseFormatFile
End With
```

The most important difference between the BulkCopy and BulkCopy2 objects is the TableLock property. If you set this property to TRUE, you will place a table-level lock on while you are executing the bulk copy. The default value is FALSE, so you must explicitly set it to TRUE to take advantage of it:

```
Dim objBulkCopy2 As SQLDMO.BulkCopy2
    Set objBulkCopy2 = New SQLDMO.BulkCopy2
    objBulkCopy2.TableLock=True
```

In addition to being able to set the properties of a BulkCopy object, you can also take advantage of its events and methods. In order to receive information from its events, you must declare the BulkCopy object at the module level with the WithEvents keyword, like this:

```
Private WithEvents mobjBulkCopy As SQLDMO.BulkCopy
```

This code will give you access to two events, BatchImported and RowsCopied. BatchImported fires at the completion of a batch and passes the user a message. You can examine the contents of the message using a message box in the event to display it in this way:

```
Private Sub mobjBulkCopy_BatchImported(ByVal Message As String)
```

```
     MsgBox Message
End Sub
```

The other event that is available is RowsCopied. In order to receive information from this event, you must be importing data using client-side BCP, and you must have set the number of rows per batch. Ensure that you are using client-side BCP by setting the UseServerSideBCP property to FALSE. You can set the number of rows per batch by setting the ImportRowsPerBatch to a cardinal value. This code sample imports data into Products2 and provides information back through the RowsCopied event:

```
With mobjBulkCopy
   .FormatFilePath = "C:\prod.fmt"
   .DataFilePath = "C:\prod.tbl"
   .DataFileType = SQLDMODataFile_UseFormatFile
   .UseServerSideBCP = False
   .ImportRowsPerBatch = 10
End With
Set objProducts2 =
objSQLServer2.Databases("Northwind").Tables("Products2")
   objProducts2.ImportData mobjBulkCopy
Set objProducts2 = Nothing

Private Sub mobjBulkCopy_RowsCopied(ByVal Message As String, ByVal Rows
As Long)
  MsgBox Message & " : " & Rows
End Sub
```

In addition to getting information from events, you can use the methods of the BulkCopy objects in the events. Abort is the primary method that that is used with BulkCopy objects. You can invoke it by placing instructions in BulkCopy events that will utilize it. You could use Abort in the RowsCopied event to halt an import, like this:

```
Private Sub mobjBulkCopy_RowsCopied(ByVal Message As String, ByVal Rows
As Long)
  If Rows>100 Then
    mobjBulkCopy.Abort
  End If
End Sub
```

The BULK INSERT Command

The BULK INSERT statement is a T-SQL statement that can copy data files into the database, much like BCP. Unlike BCP, the BULK INSERT statement cannot export data from the database to a file. Another difference between BULK INSERT and BCP is that BULK INSERT is faster at loading records than BCP because the BULK INSERT statement is executed in the process space of the MSSQLServer service.

The following can be done in parallel:

- Set the database to Bulk-Logged Recovery
- Ensure that the table has no indexes
- Specify the TABLOCK hint

Using BULK INSERT

We can use the BULK INSERT command to add records to the Products2 table, much as we did with BCP. If we use the prod.tbl file that we created with the –c switch, we know that the record fields are separated by tabs (/t) and that the records are delimited by (/n) new lines. With that knowledge, we can construct this statement:

```
BULK INSERT Northwind.dbo.Products2
  FROM 'C:\prod.tbl'
  WITH
    (
       FIELDTERMINATOR = '\t',
       ROWTERMINATOR = '\n'
    )
```

Like BCP in its basic configuration, the ProductID column is getting its value from the IDENTITY on the column. If we had wanted to use the values for ProductID that are in the file, we would have included KEEPIDENTITY argument in the WITH clause, like this:

```
BULK INSERT Northwind.dbo.Products2
   FROM 'C:\prod.tbl'
   WITH
     (
        FIELDTERMINATOR = '\t',
        ROWTERMINATOR = '\n',
        KEEPIDENTITY
     )
```

As with the BCP utility, you can also use format files with the BULK INSERT statement, like this:

```
BULK INSERT Northwind.dbo.Products2
       FROM 'C:\prod.tbl'
       WITH (FORMATFILE = 'c:\ prod.fmt')
```

Although we have touched on the most important arguments for the BCP utility, there are more. All the arguments that can be used with the BCP utility are listed in Table 9.8 .

Table 9.8: Arguments Used with BCP

Parameter	Values	Description
BATCHSIZE	batch_size	Sets the number of rows in a batch.
CHECK_ CONSTRAINTS		Forces the application of the table constraints during the insert.
CODEPAGE	ACP | OEM | RAW | code_page	Sets the code page for the columns with text values.
DATAFILETYPE	char | native | widechar | widenative	Allows you to set the default for the data file.
FIELDTERMINATOR	field_terminator	Allows you to specify the delimiter for the fields.
FIRSTROW	first_row	Indicates which row of the file should be the starting point for the bulk copy, used when you want to import a section of a file.
FIRE_TRIGGERS		Allows you to have the table triggers fire during the import.
FORMATFILE	format_file_path	Allows you to specify a format file to describe the data file.
KEEPIDENTITY		Allows you to override the IDENTITY values.
KEEPNULLS	KILOBYTES_PER_BATCH	Allows you to have NULL assigned to a column in place of an empty value.
kilobytes_per_batch		Allows you to define a batch on the basis of a maximum number of kilobytes.
LASTROW	last_row	Allows you to specify an ending row in your file. This is used when you want to import a section of a file.
MAXERRORS	max_errors	Allows you to establish a maximum number of errors before an import is imported.
ORDER	({ column ASC | DESC }…n)	Allows you to indicate the order of the data, if it is ordered.
ROWS_PER_BATCH	rows_per_batch	Allows you to define a batch on

Table 9.8: Arguments Used with BCP

Parameter	Values	Description
		the basis of a maximum number of rows.
ROWTERMINATOR	row_terminator	Allows you to specify the delimiter for the rows.
TABLOCK		Allows you to issue a table-level lock while executing an import.

Choosing a Data Import and Export Method

Many different factors play into what method you choose to use for importing and exporting data. Some methods can utilize only particular data stored in certain ways, whereas others can import data that are stored in a vast range of storage mediums. Others can export data, but some cannot. In order to choose the correct tool, you must carefully study your task at hand and choose the best solution.

The first step that you must take in choosing the best method is to define the requirements of the import/export job. What does it need to accomplish? How important is speed? How important is the robustness of the import solution? Will you import data, or will you export data?

Import/Export Job Requirements

Before you import or export any data, you should take some time to perform a simple analysis of the task in order to choose the best method to handle it. As you can see in Table 9.9 , all the methods have advantages and disadvantages. To make the right choice, you need to ask yourself several questions. Are you going to have to perform the task repeatedly? Do the data need to be manipulated, or can they be moved to the destination in their current form? Is it critical that the task be performed as fast as possible?

Table 9.9: Export Method Comparisons

	DTS	**BCP**	**BulkCopy SQL-DMO**	**BULK INSERT**
Performance	Normal	Good	Good	Best
Import text	Yes	Yes	Yes	Yes
Export text	Yes	Yes	Yes	No
Import OLE DB	Yes	No	No	No
Export OLE DB	Yes	No	No	No
Robustness	Best	Normal	Good	Normal
Graphical user interface	Yes	No	No	No
Programmatic interface	Yes	No	Yes	No
Transform data	Yes	No	No	No

In addition to understanding the task at hand, you need to have a clear view of the resources available. What are the skills of the people who are doing the import or export? Some people could be very good at using BCP, whereas others might be quite proficient at using DTS. Are there packages or format files that already exist that you can take

advantage of? Once you know what you have to do and the resources that are available, choosing the best method to accomplish it is trivial.

Another consideration that you must undertake when choosing the vehicle for moving data is whether you are exporting data or importing data. The BULK INSERT statement only imports data and has no avenue to export data. The BCP application, BulkCopy objects, and DTS all can import and export data.

Existing Data Format

One of the considerations that you must take into account is the format of the source and destination data. Are the data in a SQL Server database, an Oracle database, a spreadsheet, a text file, or some other format? The data format matters quite a bit because some tools can work with only particular formats. The BULK INSERT statement can work only with files as its source and a SQL Server as its destination. The BCP application and BulkCopy SQL-DMO objects can deal with files as either the source or destination but must have a SQL Server table and view as the other.

DTS has a great deal more flexibility and not only can deal with SQL Server entities and text files but also with any data that can be accessed through OLE DB. This includes data sources such as Excel, Oracle, and any data source that supports the OLE DB provider for ODBC. In addition, SQL Server does not need to participate as it does with a BCP application, BulkCopy SQL-DMO, or the BULK INSERT statement.

Frequency of Import or Export Task

In addition, the frequency with which you must import or export a certain type of data can have an impact on which method you choose. If you need to do something multiple times, you would probably choose a more robust programmatic tool. In fact, many people believe that if you do anything more than once, you should write a program to do it. DTS packages and the BulkCopy SQL-DMO objects allow you to build programmatic solutions that in turn allow you to build robust import and export applications suited to repeat the same task over and over again.

Even though BCP applications and the BULK INSERT statement are better suited to quick, one-time-only uses, there are things that you can do to make them easier to work with on repetitive tasks using format files. Format files help when you are repeating the same data import using the BCP application and the BULK INSERT statement and the exports using the BCP application. However, because the BCP application is a command-line utility and the BULK INSERT statement is a T-SQL statement, they are less suited for high-frequency tasks; they offer less robust error handling and less programmatic flexibility.

Data Manipulation Tasks

The only true option for manipulating data as you import them is to use DTS. You can massage data types with BCP, the BulkCopy SQL-DMO objects, and the BULK INSERT statement, but you cannot apply business logic to the data as you can with DTS. DTS was designed to move great amounts of data and to transform them programmatically.

Performance Considerations

Speed is often a very important consideration in any system, and performance differs greatly among the potential methods. Often, speed comes at the expense of robustness. Generally, the more complex a tool, the slower it is. DTS packages are slower than an application using the BulkCopy objects. An application using BulkCopy objects is generally slower than the BCP command-line application. Likewise, it is important to note that the BULK INSERT statement is much faster than BCP or the data pump. You can take advantage of this in DTS by using the Bulk Insert task, because the Bulk Insert task uses the BULK INSERT T-SQL statement. However, you must remember that the BULK INSERT T-SQL statement cannot perform transformations. The more layers and sophistication a data movement method uses, the more system resources will be consumed.

However, there are reasons that some solutions perform better than others, and those lessons can be applied to more robust solutions. One of the reasons that BULK INSERT has a higher throughput is that it is executed in the process space of the MSSQLServer service. A DTS package, a BCP command, or an application built with the SQL-DMO BulkCopy objects can also be executed in the process space of the MSSQLServer service, but it will also include the extra overhead it takes to run the package, command, or application.

No matter what data import strategy you use, dropping all the indexes on a table will greatly increase the performance of your import. When you recreate the indexes, the clustered index on the table should be created first, before you create the nonclustered indexes. If you want to recreate the indexes in parallel, you can, but you must first create the clustered index, then execute the recreation of each nonclustered index from a different connection. This might be easier to accomplish with DTS because you can build a task that will tear down the indexes, store their structure, and then rebuild them after completion. You could also accomplish this task with BulkCopy SQL-DMO objects, but it would be more difficult to do with either the BULK INSERT T-SQL of the BCP application. In addition, setting the database to Bulk-Logged Recovery will allow the operation to proceed faster because the inserts will not be logged. In addition, this method helps you avoid worry about filling up the transaction log.

Summary

In a world of data warehouses and disparate data sources, importing and exporting data are two of the most important tasks that a database administrator must do. Because of their significance, Microsoft has included a number of tools that expedite the flow of data—not only between SQL Servers, but also between a wide array of heterogeneous data sources.

Data Transformation Services (DTS) is the most powerful data-movement tool included in SQL Server. With it you can transfer data between any data sources that have OLE drivers. In addition to native OLE DB drivers, all the data sources that can be accessed through ODBC are also accessible through the OLE DB ODBC driver. Even though DTS is a SQL Server tool, SQL Server does not have to be the source or destination data source.

Included with DTS are the DTS Designer and DTS Import/Export Wizard. With these tools, you can create packages, which are the encapsulation of the tasks, connections, transformation, and presentation information needed by the DTS runtime to perform data import and export. Although both the DTS Designer and DTS Import/Export Wizard can create packages, the DTS Designer is the principal tool to edit packages. Packages can be saved to a variety of formats, including to tables in a SQL Server, SQL Server Meta Data Services, a structured format file, and a Visual Basic file.

Just as there are many ways to save a package, there are many tools to execute a package. DTS is not the only solution to import and export data; you can also use the Bulk Copy Program (BCP) application and the BulkCopy SQL-DMO objects. The BCP application and the BulkCopy SQL-DMO objects are legacy tools that allow data interchange between a file and a SQL Server. They can allow data to flow either way. The BCP application is a command-line utility that is not very robust, but a great number of people are greatly skilled with it. The BulkCopy and BulkCopy2 SQL-DMO objects are programmatic objects that can accomplish the same interchanges as the BCP application, but they are more robust due to their programmatic nature. You can also use the BULK INSERT statement to import data. It is the fastest import method, but it cannot be used to export data.

Choosing a data import method is very important in establishing your data movement solutions. DTS is the most vigorous method, making it an excellent choice for your data-movement needs. The BCP application is a command-line application that can transfer data between a text file and SQL Server. It has been around a long time and, as a consequence, a large pool of people know how to use it effectively. The SQL-DMO BulkCopy and BulkCopy2 objects provide the same functionality as the BCP application, but they can be encapsulated in an application providing a robust interface suited for tasks that need to be executed repeatedly. Finally, the BULK INSERT statement can insert records faster than any other method, but it cannot export data. Each method has its advantages and disadvantages. You must choose the best method based on the requirements of the task and the resources available.

FAQs

Q. How would I create a Visual Basic subroutine that would execute a package file using a trusted connection?

A. You would need to write a routine similar to this:

```
Private Sub RunPackage(ByVal PackageName As String, _
                       ByVal FileName As String)
Dim objPackage      As DTS.Package2
Dim objStep         As DTS.Step
Dim objTask         As DTS.Task
Dim objExecPkg      As DTS.ExecutePackageTask

    'Instantiate package
    Set objPackage = New DTS.Package
    objPackage.FailOnError = True

    'Create the Step, Task and ExecutePackageTask
    Set objStep = objPackage.Steps.New
    Set objTask = objPackage.Tasks.New("ExecutePackageTask")
    Set objExecPkg = objTask.CustomTask

    'Specify the package and security information
    With objExecPkg
        .UseTrustedConnection = True
        .FileName = FileName
        .Name = PackageName
    End With

    'Link the Steps to the Tasks
    With objStep
        .TaskName = objExecPkg.Name
        .Name = PackageName
        .ExecuteInMainThread = True
    End With

    'Link the Steps and Tasks to the package
    objPackage.Steps.Add objStep
    objPackage.Tasks.Add objTask




    'Run the package
    objPackage.Execute

    'Release objects
    Set objExecPkg = Nothing
    Set objTask = Nothing
    Set objStep = Nothing

       objPackage.UnInitialize

End Sub
```

Q. If I use the SQL-DMO BulkCopy object, how can I get the total number of records

that I have imported or exported without using events?

A. Both the ExportData and ImportData methods return a value that indicates the number of rows copied. You need to place the BulkCopy object in parentheses and set the value of the method call equal to a variable. The variable contains the number or rows copied. Here is an example illustrating what you would need to do:

```
Dim lngNumberOfRows as long
lngNumberOfRows = objProducts2.ExportData(objBulkCopy)
```

Q. If I had no infrastructure in place, what would be the best Microsoft-based data transfer technology to invest our resources into learning how to use?

A. Clearly, DTS is the tool of today and the future. It offers a robust, interactive development environment that allows you to work with a variety of data sources exchanging, splitting, consolidating, and transforming data between them. None of the technologies offers these advantages. The only negative with DTS compared with other Microsoft data-movement solutions is speed, but that is a small price to pay for the system's robustness nature.

Q. Most of the database administrators who work for me are very good at using the BCP application; should I move to using DTS instead?

A. If your people are proficient at using BCP and that encompasses all your needs, you should continue to use it. However, you should start making some investment in learning how to use DTS, because it can be said that DTS is the future, whereas BCP is the past. DTS will continue to grow, adding new abilities and refinements, whereas BCP has been and most likely will continue to be stagnant.

Chapter 10: SQL Server Analysis Services

Introduction

Analyzing and understanding your data are the roles of SQL Server 2000's enhanced OLAP and new data-mining components. Renamed *Analysis Services* in SQL Server 2000, online analytical processing (OLAP) services have been enhanced to provide greater capability and scalability. Enhancements such as distributed partitioned cubes, indexed views, dimension enhancements, real-time OLAP, and security enhancements are only a few of the changes that have shaped the new Analysis Services for SQL Server 2000. New tools and wizards break down some of the previous barriers that have kept smaller organizations from taking advantage of OLAP for decision support and analysis. Creating cubes, fundamental components of OLAP, is now possible using step-by-step wizards as well as the cube editor utility for advanced users. Enhancements to Multidimensional Expressions (MDX) and the new MDX Builder utility assist users in querying OLAP solutions and discovering the data without complete understanding of MDX syntax.

An exciting addition to SQL Server 2000 is the new data-mining capability in Analysis Services. Data mining in SQL Server 2000 allows organizations to discover data relationships and trends in both their relational data and OLAP cubes. Understanding data relationships can help organizations make informative predictions of future activity and respond to those trends. This is fundamental to many organizations such as ecommerce and traditional retail companies, which can benefit from a better understanding of users, purchase trends, marketing response, and more. The possibilities for an accessible data-mining solution are endless, and many organizations that have not been able to take advantage of this technology due to the complexities in traditional data-mining solutions will find it critical to their success in the future.

All these enhancements are complemented by their accessibility. SQL Server 2000's support for HTTP/HTTPS access and integration with Windows 2000 Active Directory allow the location and use of these services across the enterprise. Remote clients can connect to OLAP and data-mining solutions using HTTP/HTTPS over the Internet for querying and analysis. Large organizations can publish analysis servers in Active Directory, allowing users to locate and use the capabilities of SQL Server 2000's Analysis Services.

This chapter reviews the enhancements and additions to SQL Server 2000's Analysis Services. The chapter's review of OLAP and data-mining technologies will assist you in understanding the capabilities and decision support power of this technology. Examples step you through configuring and querying your own OLAP cubes and using the Data Mining Model Browser to discover trends in your data.

Online Analytical Processing and Data Mining

In the early days of corporate computing, relational database systems were primarily used to capture or generate data critical to business operations. Such systems include order processing, invoicing, and customer information systems. Although without doubt the data processed differed between organizations, the systems were often characterized by the use of *transactions* . Such systems, termed *online transaction processing (OLTP)* systems, are still in use today in nearly every organization. However, over time, improvements in database engines, underlying hardware and infrastructure, and the automation of many tasks has gradually allowed a shift in focus from merely the capture of data to the analysis of that data to assist in driving business decisions. This focus has led to the development of data warehousing, which is essentially the gathering, filtering, and cleansing of organizational data so that it can be presented to an audience that wants to use it for reporting and business analysis. A term that defines the analysis of this collected data in data warehouses is *online analytical processing (OLAP)* .

Note The concept of OLAP is not a new one, but the term is, having first being associated with this technology by Dr. E. F. Codd, the father of the relational database, in a paper published in 1993.

OLTP vs. OLAP vs. Data Warehousing

Perhaps the primary goal of data warehousing is to provide data from across the entire organization in a state that can be used by users who require it to make informed business decisions. The term *data warehouse* is used to describe an unchanging store of collected data, gathered from the entire organization and available for browsing by users. You might also encounter the term *data mart* , which is a subset of data from a data warehouse, commonly data for a single department.

So what is the relationship between OLAP and data warehousing? Are the two terms interchangeable? The short answer is no. A data warehouse is a database containing (in the majority of cases) historical data from an organization stored in such a way as to aid analysis, whereas OLAP provides the mechanism to analyze this data by providing the definition, processing, and delivery of analysis objects. Consider the concept of data warehousing to encompass the organization and arrangement of a company's data for storage, and OLAP to be the means by which consumers are able to access the data. OLAP is used to serve up data to consumers using natural, intuitive data models, thus allowing better navigation and understanding of the collected information. Well-designed OLAP solutions should meet the FASMI test, outlined below:

- **Fast** The requested information should be delivered to the end user at a consistent rate. The aim is to have the majority of queries complete within five seconds.
- **Analysis** The solution should be able to perform rudimentary numerical and statistical analysis of the data, whether it is determined at design time by the developer or at runtime by the user.

- **Shared** The solution should implement the required security measures to ensure that confidential data can be shared across a large user base.
- **Multidimensional** The data should not be considered simply as rows and columns; it should also be given other dimensions, such as time periods. Furthermore, it must support enough hierarchies and levels to accurately model the data.
- **Information** All required data should be accessible by the solution, regardless of the data's location and without volume restrictions.

In a nutshell, OLAP allows quick and easy access to shared multidimensional information, allowing non-DBAs to be productive data analysts. However, this access has one very important proviso: There must be a means of gaining access to the multidimensional data. Within Analysis Manager, this access is provided as part of the package; however, from outside that tool, access to multidimensional data is not necessarily easy unless an analysis application has been developed. The means of providing access from client tools and applications is to use MDX for querying data warehouses and providing the data to the client application for use. This is a relatively new area in the SQL Server world and as such is unfamiliar, even to many DBAs. We discuss MDX expressions later in this chapter.

If you have been around database solutions for some time, you might ask why we need OLAP. After all, OLTP solutions still provide reporting capabilities. Perhaps the best way to illustrate the advantages of OLAP to an organization is to contrast it with traditional reporting. Such reports offer a snapshot of the underlying data at a point in time, whereas OLAP techniques provide a series of guided shots in which the analyst controls the position of each shot. OLAP allows you to move through the data, drilling down into greater detail and following interesting trends. Traditional reporting methods could suffice for some decision-making processes, but OLAP methods empower the analyst and provide greater, more complete access to useful information. Data warehousing also typically contains much more historical data than is active in an OLTP system. Large-scale analysis and trending are difficult with time-limited OLTP data.

OLAP can be used for either preanalysis or postanalysis exploration of data. Preanalysis is used to find particular trends or exceptions among the data; post-analysis exploration is used to investigate these trends once they are uncovered. Thus, OLAP tasks may be split, allowing some users to perform high-level analysis, before passing the results on to area specialists for more detailed probing.

OLTP systems and data-warehousing systems have different goals; hence different items are important to each type of system. OLTP concentrates on performance in order to be able to capture data as quickly as possible and be ready to continue operation. In general, OLTP systems are characterized by a large user base (often numbering into the thousands) resulting in continual use of the read and write operations by an ever-changing subset of those users. Transactions made in OLTP systems are typically short and the size of the data is kept as small as possible in order to allow high throughput of data (giving greater performance). To picture the demands on an OLTP system, think of

the last time you visited the supermarket. Each of the many checkout aisles had an operator capturing transactional data, in short bursts, about each individual product in each person's purchase. This is a small-scale OLTP system, but when you picture that store as only one in a large chain, all of which ultimately feed a large database, you can begin to imagine some of the challenges involved in keeping such a system running smoothly.

Performance and availability are crucial to OLTP systems, but they are not as important to data-warehousing systems. However, data-warehousing systems do have their own set of requirements and properties. Because the data in data warehouses is rarely modified, such systems have infrequent write operations, except during data load periods, when writes occur in massive amounts of data as the warehouse is updated. However, such systems provide a read-only interface to a small user audience, with an even smaller number of simultaneous users. The information in data warehouses has most likely come from a number of sources and been made consistent (you might hear this process referred to as *data cleansing*) before it is presented to the user, so although OLTP systems are often available and used 24 hours a day, seven days a week, data-warehousing systems cannot be used until the data have been correctly prepared. These processes of data assembly and cleansing often result in the data in the database having a far larger size than that seen in OLTP systems.

Another key difference between OLTP and data-warehousing systems relates to their design. OLTP databases make greater use of the rules of normalization, with most databases in the third normal form. This form aids in keeping transactions short but provides a major hindrance to data-warehousing systems because they often work with huge amounts of data, and performing the multitude of required joins to extract meaningful data can be slow, even on systems with the biggest and best hardware configurations. For this reason, the data in data warehouses is not stored in a highly normalized form. In fact, data warehouse designers don't use the entity relationship (ER) model to design databases. Instead, they use a logical model known as the *dimensional model.* Each dimensional model is composed of one fact table, which stores information such as sales or profit data, and a number of other tables that store dimensions such as time or geographic location. Such tables are often arranged in a snowflake structure, as shown in Figure 10.1 .

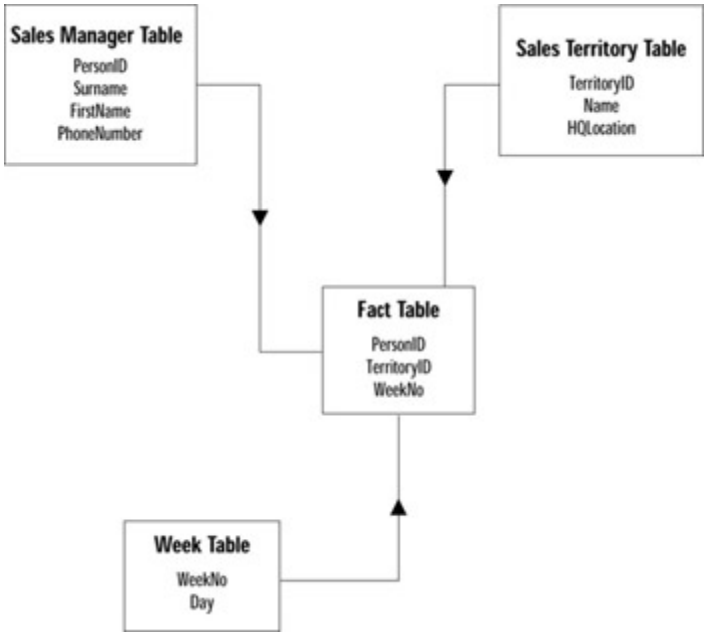


Figure 10.1: The dimension model illustrating a star schema.

Note The third normal form describes databases in which there are no redundant data (in other words, data are not stored more than once within the database) and no data are dependent on nonkey fields. However, few production databases are fully normalized. Normally, some degree of denormalization is used in order to receive performance gains. This, however, does not suggest that a database should be designed denormalized with performance in mind. Always fully normalize a design, and denormalize only when you have proven it is required.

Data Mining

We've discussed the concepts behind OLAP; from here it's just a short hop to understanding data mining. Put simply, data mining is the process for analyzing information in order to discover among the data trends or other interesting highlights that are not visible through data browsing. Various algorithms can be used to mine the data. Currently, SQL Server 2000 supports two built-in algorithms: decision trees and clustering.

Decision trees classify information in a tree-like structure. Each branch junction in the tree represents a question, with the answer determining which path is followed and hence which question will be posed next. This structure can be used to help you predict likely values of an attribute of the data. Take, for example, a bank employee presented with the task of analyzing an application for a home loan. The employee could use a decision tree to determine if the bank customer is likely to be able to repay the home loan and therefore if the loan should be approved.

Clustering groups heterogeneous data into several homogenous subgroups, or clusters. If that statement sounds like just so much jargon, think of a cluster instead as grouping of like data together from a mass of dissimilar data. There are no set criteria for grouping

the data; similar data are assimilated, and the meaningfulness (or otherwise) of the groupings is left to be determined by the analyst. As an example, consider a national sales organization that plans to open a new territory and wants to direct its advertising budget into the section of the community in that territory most likely to buy from its product range. A clustering analysis could assist the organization facing this question. By feeding demographic and sales information from existing sales territories and demographics from the new territory into a data-mining solution, the organization can generate a model to suggest the most likely areas in which sales can be generated, and then advertising dollars can be channeled into the areas in which greater return should be achieved.

New Features in Analysis Services

SQL Server 7.0's OLAP Services has been renamed Analysis Services in SQL Server 2000. Given the name change, you might well expect significant changes to the product, and the product team has delivered on those expectations, introducing a number of new features and providing enhancements to others. Let's briefly look at the most significant new features in Analysis Services. Many of these features are examined in more detail later in this chapter.

OLAP Enhancements

The following sections describe OLAP enhancements to SQL Server 7.

Cubes

At the core of SQL Server's OLAP capabilities is the use of *cubes* , so it is not surprising that a number of enhancements have been made to OLAP cubes, significantly increasing both their functionality and ability to scale. It is now possible to use cubes for real-time OLAP, which uses *relational OLAP (ROLAP)* to allow constructing multidimensional views of data that are dynamic. With this type of technology, it is now possible to analyze short-term trends in fast-changing data.

Storage Locations

Cube data can be stored across a number of analysis servers using the new distributed, partitioned cube facilities. It is also possible to spread a number of cubes across various analysis servers and access them all by connecting to only one of those servers. This is *linked cube technology,* and perhaps its biggest benefit is to reduce storage costs because now each cube needs to be stored only once (on a single server).

Actions

The concept of actions was introduced earlier. *Actions* are sets of operations that have been defined in advance. You, as a user, are able to invoke them against the cube or just a portion of it during analysis. These actions are even able to perform external tasks such as launching and passing parameters to applications. Consider, as an example, an analyst looking at sales data across various cities for several product lines. Noticing that sales of a particular product are low in a particular city, the analyst invokes the Discount action, which cause price changes to occur within the Sales and Merchandising systems.

Access from Client Applications

The newest version of SQL Server also includes enhancements for the users who browse cubes with client applications. As long as the client application supports drill-through, users are now able to drill down from a cube cell and work with result sets from the source data for the cell, allowing much more detailed analysis than was previously

possible. OLAP designers can now also hide entire cubes or merely elements or cubes, such as levels or measures, from various client users, thereby allowing a more customized and streamlined solution for the end user.

Dimensions

A number of developments have been made in the area of *dimensions* as well. Several new dimension types are now available. These include *ragged dimensions* , which allow the logical parent of an item to reside outside the next level in the hierarchy. For example, a ragged dimension allows a front-line employee's immediate superior to be the managing director of the company, even if a level of middle management exists. Analysis Services also introduces the new *parent/child dimension* type, which uses two columns of the dimension table to outline the relationship between its members—for example, one column could identify an employee's payroll number and the other could identify the payroll number of that employee's manager (that is, the person directly above the employee in the hierarchy).

The ROLAP storage mode allows larger dimensions to be catered for than previously possible. It is now possible to change dimensions without the need to fully process the cube before using it for analysis. This feature gives higher availability of the data for analysis and results in a better quality of information when the underlying data frequently change. Furthermore, write-enabled dimensions have been introduced, allowing users to make changes to dimensions and see immediate impact on the cube.

Warning Although you might be tempted to use changing dimensions frequently due to their advantages, bear in mind that their flexibility comes at a cost; queries using changing dimensions perform more slowly than those using unchanging dimensions.

Security

Security for Analysis Services has also been enhanced, now with greater flexibility with regard to access to cube data. Access to cubes can now be controlled down to the cell level, as well as determining the access to grant to dimensions, levels, and members. In addition to more traditional authentication methods, clients are now able to connect to, authenticate, and use analysis servers via HTTP and HTTPS.

Integrating the Add-Ins

The SQL Server team has also made the DBA's life easier by integrating a number of products that were add-ins for SQL Server 7.0 OLAP Services. For example, the Archive and Restore Database options are now fully integrated with Analysis Services. In addition, an MDX Builder tool is available to allow MDX statements to be generated using drag-and-drop techniques. Using the MDX Editor is now easier, thanks to the introduction of code-coloring techniques and tool tips to deliver syntax prompts. If

you've used other Microsoft development tools such as Visual Basic, C++, or even Access, you'll be familiar with these nice touches.

Data-Mining Capabilities

SQL Server 2000 is the first version of the product to provide support for data mining. Although this component of Analysis Services is in its infancy, it is still a major step. Not only does it provide support for developing data-mining models using either clustering or decision tree algorithms, but it also provides support for the OLE DB for Data Mining API, which allows third-party providers of data-mining algorithms to integrate their products with Analysis Services, thereby further expanding their capabilities and reach.

The Analysis Services Architecture

Microsoft Analysis Services is made up of a number of components spread across the server and the client. Figure 10.2 illustrates the architecture of Analysis Services and interaction between the tiers.

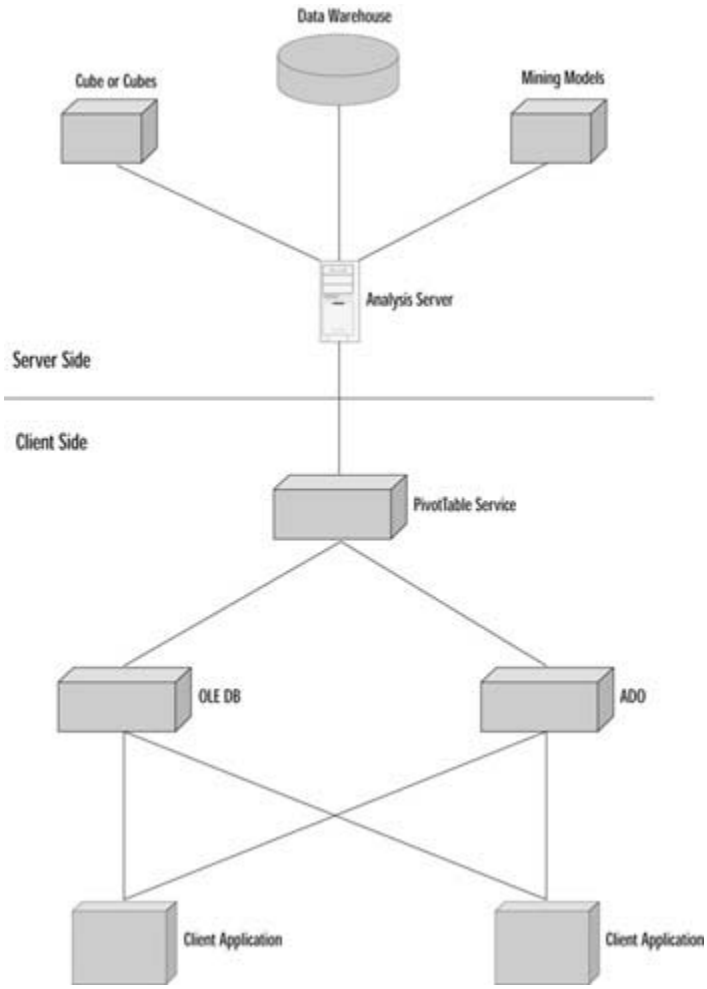


Figure 10.2: The Analysis Services architecture.

Analysis Server

Analysis Server is the heart of Analysis Services; it processes cubes and sends the results to the client. Therefore, at least one Analysis Server must be running if you want to analyze data using client software. More than one Analysis Server can be set up and can contain one or more databases, and it must be registered if you want to administer it using Analysis Manager. Analysis Server runs as a service, and like other services, it can be monitored, started, and stopped via the Control Panel. Should you want to perform these actions, look for the MSSQLServerOLAPService.

Each Analysis Server has a repository—the Analysis Services repository, which holds the definitions of the objects defined on the server. By default, the Analysis Services repository is given the name Msmdrep.mdb and stored on the computer running the Analysis Server. Each Analysis Server has an associated Data folder in which to store the structures for the objects on the Analysis Server. It is these structures that are used in the processing of queries sent to the Analysis Server.

Analysis Manager

Analysis Manager is the primary tool by which you will access Analysis Server. Like many of Microsoft's server-based tools, Analysis Manager is a Microsoft Management Console (MMC) snap-in that results in an easy-to-use and intuitive interface. If you've used SQL Server 7.0 before or if you've used SQL Server 2000 for a while now, you'll find the interface has a look and feel similar to those of Enterprise Manager. Analysis Manager presents a node for each Analysis Server and a subnode for each database contained within the Analysis Server.

Five folders are contained under each database node. They are:

- **Data Sources** This folder contains information related to the server connection, security, and OLE DB provider. New data sources can be set via a mouse right-click from this section; this operation is covered in more detail later in this chapter.
- **Cubes** This folder holds all the cubes that belong to the database. In Figure 10.3 , you can see that the FoodMart sample database has a number of cubes available, the first three of which are Budget, HR, and Sales. Within cubes are the Partitions and Cube Roles subfolders, covering the partitioning and security issues of the relevant cube. The specifics of these items are covered later in this chapter when we build a cube together.

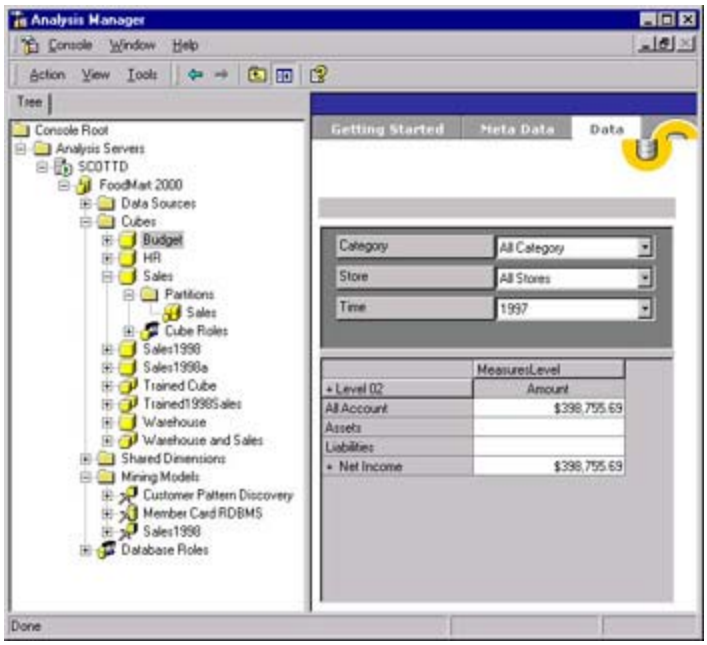


Figure 10.3: Analysis Manager, showing an expanded tree.

- **Shared Dimensions** This folder contains the dimensions shared between two or more cubes. A common shared dimension is time.
- **Mining Models** This folder contains models based on cube information built using the Mining Model Wizard.
- **Database Roles** This folder contains security accounts that are used to grant and restrict access to users attempting to use Analysis Services components such as cubes and mining models. Roles in general contain Windows user accounts or groups, but may also contain SQL Server logins or other roles.

Cubes

Information within OLAP models is seen conceptually as *cubes* . Cubes are multidimensional structures containing an organized subset of data from a data warehouse. Each cube consists of dimensions that are essentially descriptive categories, such as time or geographic location, and *members* , which are values such as unit sales and population.

Within cube dimensions, data are organized into hierarchies in which member data are grouped together into what essentially amounts to levels of data, such as days, hours, and minutes within the time dimension. Data consumers move up or down through the hierarchy, depending on the level of detail they require. Hierarchies are defined in terms of such levels, where the highest level contains the most summarized data and the lowest the most detailed data.

Within cells, we can identify specific information by the coordinates at which the information resides. Each cell is a set of information that takes one member from each dimension. For example, if we are analyzing product sales by territory across time, we

could set up a cube with three dimensions (product, time, and territory) and would have potential cells as follows:

- Widget, January 2000, West Coast
- Sprocket, March 2000, Central Highlands

Note Although we traditionally think of a cube as being a three-dimensional structure, it might surprise you to know that within SQL Server, a multidimensional cube can actually contain up to 64 dimensions!

Cubes in Analysis Services support a number of new or enhanced features. (Space limitations prevent complete coverage here; refer to Books Online for this information.) Among these features are the following:

- **Calculated cells** These allow the use of formulas containing conditional calculations for single cells or groups of cells to store different data in different scenarios.
- **Real-time OLAP** This feature allows cube data to be continually updated as the base data changes, hence allowing OLAP solutions to be used in solutions in which data are highly dynamic or gathered at a rapid pace.
- **Enhanced cube processing** This feature allows users to access and use cube data while aggregations are still being calculated.
- **Hidden cube elements** These allow certain aspects of cubes, such as dimensions or measures, to be made invisible to selected end users browsing cube data from client applications.
- **Drill-through** This feature allows users browsing cube data to drill down on a particular piece of data and access the record set that comprises the source data that made up that cell.

Mining Models

Data mining models are core to the concept of data mining and are virtual structures representing data grouped for predictive analysis. At first glance, mining models might appear to be very similar to data tables, but this is not the case. Tables are used to represent actual collections of data, whereas mining models are interpretations of those data, known as *cases* . These cases store statistical information representing the rules and patterns learned from training the model.

The model is trained by feeding existing information and trends to it; we cover this concept in more detail later in this chapter. Cases are grouped to together to form *case sets* , which make up a mining model.

A data-mining model is structurally composed of a number of data-mining columns and a data-mining algorithm. The content created when the model was trained is stored as data-mining model nodes. It is important to realize that the data used to train the model are not stored with it; only the results are stored. This structure enables extremely large volumes

of data to be used during the training process, thereby (hopefully) increasing the likely accuracy of any predictions made by using the model.

- **Data-mining columns** These define the inputs to and outputs from the mining model. The columns can be used with familiar SQL syntax to either add training data (with INSERT statements) or query the predictive results during the analysis phase. Each column can contain a solitary data item, such as an integer, or alternatively, a group of other data-mining columns. This system allows hierarchical situations to be modeled. Data-mining columns are accessed using either the Relational Mining Model Editor or the OLAP Mining Model Editor.
- **Data-mining algorithms** These determine how the mining columns should be processed. They supply the decision-making logic and are frequently based around a particular goal, commonly to determine the likely value of an attribute or attributes in a given scenario. Data-mining algorithms are supported and implemented via a data-mining algorithm provider. Analysis Services currently supports two providers: Microsoft Decision Trees and Clustering. Both of these items are covered in more detail later in this chapter
- **Data-mining algorithm providers** These are simply OLE DB providers that support the OLE DB for Data Mining specification., so it is likely that we will see more algorithm providers appear in the future, supplied by either Microsoft or third-party vendors. Such providers are expected to either expand on the capabilities of the current providers or use another type of data-mining technique.

PivotTable Service

PivotTable Service is the main interface for applications that use Analysis Services. It is an OLE DB provider for data-mining operations and multi-dimensional data access. PivotTable Service allows client applications to browse, analyze, and update data from cubes managed by an Analysis Server. PivotTable Service even allows you to store data in a cube on a client computer. This last feature is available because PivotTable Service functions as a local multidimensional data server. This duality also provides other benefits: Some data analysis can be performed at the client without the need to connect to an Analysis Server, and results can be cached on the client, helping to keep traffic down and provide performance improvements.

PivotTable Service can be used from a number of development platforms, including Microsoft Visual Basic or Visual C++, using either the ActiveX Data Objects Multidimensional (ADO MD) object library or the OLE DB for OLAP Component Object Model (COM) interface. Detailed discussion of the development of such applications is outside the scope of this book. The PivotTable Service has acquired some interesting new capabilities. Among them are:

- **Internet connectivity** It is now possible for client applications using the PivotTable Service to gain access to Analysis Services across the Internet using HTTP. More detail on the specifics of this feature is provided later in this chapter.

- **Data-mining support** PivotTable Service has been extended to support the new data-mining capabilities of Analysis Services. Both Microsoft Decision Trees and the Clustering algorithm are supported.
- **Altering cube structure** The ALTER CUBE statement is now supported by PivotTable Services, thereby allowing client applications to make changes via Data Definition Language (DDL). Such changes could include changing or adding dimensions members, moving dimension members, or removing a dimension member.

Decision Support Objects

Decision Support Objects (DSO) is a library of classes and interfaces that provide access to an Analysis Server. DSO models the basic storage mechanisms and elements used by Analysis Services, allowing them to be controlled programatically by any language able to use Microsoft's COM technology. These basic elements are represented in a hierarchical structure and are databases, data sources, dimensions, cubes, data-mining models, and roles. The server object sits at the top of the hierarchy.

Installing Analysis Services

You've now read about a number of the new features and capabilities of Analysis Services and are no doubt ready to roll up your sleeves and start to get some hands-on experience with the product. Before you do, however, we need to install the software. This is not a difficult process, but remember that we are dealing with an enterprise solution and a tool that could well be used to help make mission-critical decisions. Therefore, we need to ensure that the installation process is carried out correctly and in full.

Analysis Services Requirements

Before you begin to install Analysis Services, make a basic check that the target computer meets the minimum requirements for installation. Check the specifications of your machine against the following list:

- Intel or compatible computer with either a minimum of a Pentium 133MHz processor, Pentium Pro, Pentium II, or Pentium III.
- 32MB of RAM (a minimum of 64MB is recommended).
- A minimum of 50–90MB of free hard drive space; 130MB will be required if you want to install all the shared components and available samples, and 12MB is required for the client software.
- Windows 2000 Server or Windows NT 4 Server (with at least Service Pack 5 applied). Other OSs are supported for client-only software; full details might be found in the SQL Server 2000 Books Online (BOL).
- Built-in Windows network software from Windows 95 or NT 4 or higher. TCP/IP is required.

In addition, to successfully install Analysis Services, you must log on to the server with local administrator rights. Online help or documentation requires Internet Explorer 5 or above.

Installing Analysis Services for the First Time

Analysis Services is installed from the SQL Server 2000 CD-ROM; insert the CD into your CD-ROM drive and, after a short wait, you should be presented with the SQL Server Automenu.

Tip If you don't see a menu, it suggests that the Autorun feature is disabled for your CD-ROM drive. You can execute the Automenu by browsing the disk for a file named autorun.exe and then executing this file.

Follow these steps to ensure that Analysis Services is installed correctly:

1. Ensure that any other running software is stopped to allow the setup procedure uninterrupted access to any shared files that might need to be updated. Then select SQL Server 2000 Components from the left portion of the screen.
2. A new screen will appear, from which you should select Install Analysis Services. This action launches the Installation Wizard. You can sit back and enjoy the ride for a little while; just follow the steps of the wizard.
3. Press the Next button when prompted, then read and agree to the End-User License Agreement.
4. The next screen will ask you to choose which Analysis Services components you want to install; just accept the defaults at this stage—that's all you need to complete the exercises in this chapter. You can always install other components as the need arises. If you want to change the destination folder for the installation, do so now using the Browse button. Otherwise, press the Next button to move on.
5. Next, select the location for the data; either accept the default or browse for a new location. Whichever choice you make, you must ensure that the chosen location is secure, because this folder contains files that control users' access to Analysis Services objects. It is possible to store data on a remote computer. If you choose this option, make sure that you enter the fully qualified computer name and path. You also need to instruct the Analysis Server Service to log on as your user account, not as the (default) system account. Depending on your OS, this task can be achieved via the Services or Administrative Tools options in the Control Panel.
6. Click Next. Accept the default Start Menu Program folder, and allow Analysis Services to install. You might be prompted to restart your computer to complete the installation process. If this is the case, restart the computer before you attempt to use Analysis Services.

Warning You must make sure that the folder to which Analysis Services is installed uses only single-byte characters in its path. Installing Analysis Services to a double-byte character set could produce unwanted results.

Table 10.1 describes components that should have been installed on your system in the locations specified.

Table 10.1: Analysis Services Components and Default Locations

Component	**Default Location**
Analysis Server (To function correctly, this component requires the client components described elsewhere in this table.)	C:\Program Files\Microsoft Analysis Services\ Binn
Analysis Manager (To operate fully and correctly, this component requires DSO and the client components described elsewhere in this table.)	C:\Program Files\Microsoft Analysis Services\ Binn
DSO (To function correctly, this component requires the client components)	C:\Program Files\Microsoft Analysis Services\ Binn

Table 10.1: Analysis Services Components and Default Locations

Component	Default Location
Client components (Files such as PivotTable Service that are required by the Analysis Services client.)	C:\Program Files\Microsoft Analysis Services\ Binn
Books Online (An absolute must!)	C:\Program Files\Microsoft Analysis Services\ Help
Sample application (The FoodMart database we use in this chapter to demonstrate ideas and exercises.)	C:\Program Files\Microsoft Analysis Services\ Samples
Data folder (This component holds the data from Analysis Server.)	C:\Program Files\Microsoft Analysis Services\ Data

Upgrading from Earlier Versions

Before upgrading to Analysis Services, back up the query log (msmdqlog.mdb) and metadata repository (msmdrep.mdb) located in the \Bin directory of your Analysis Server installation. Neither the query log nor the repository database are replaced or deleted during the upgrade process. Once this is done, you can install Analysis Services from the SQL Server 2000 Components Setup option. For instructions, refer to the previous sections in this chapter.

Designing and Building an OLAP Solution

OLAP technology is without doubt a great tool for obtaining data from data warehouses. However, using it is not as straightforward as simply opening a database and browsing the data in a cube. In order to get good, usable data on a regular basis, you should look to build an OLAP solution. The next section takes you through the process of building such a solution: how to approach it, the steps required, and any pitfalls you might face along the way.

Designing and Building a Data Warehouse

Although a number of wizards exist in order to assist you in getting your data warehouse up and running, unfortunately they are only tools and are not replacements for good planning and design. The following section discusses the design concepts that you should consider prior to and during the building of the data warehouse. The section also covers the techniques required to actually build the warehouse and populate it with data.

Normalization vs. Denormalization

Unless you are very new to the world of databases and database design, you'll no doubt be familiar with the relational model and entity relationship (ER) diagrams. Relational databases are highly normalized, usually to the third normal form, in order to remove redundant data and keep transactions as small as possible. This approach works well for OLTP databases because transaction time is kept to a minimum due to the small amount of data involved. However, OLAP systems are not concerned with data throughput; rather, they involve the retrieval and analysis of data. OLAP systems often analyze large amounts of data that are both deep and wide. Such data, when stored in a relational database, would not only run to many rows, it would also be spread across many columns in many tables. In order to restructure this data for analysis, many joins would have to be processed for each row returned, resulting in very slow system performance. In order to address this problem, data warehouses use the dimensional model. This model makes no use of normalization; rather, it uses a structure made up of attributes represented across a number of dimensions.

Components of a Data Warehouse

A data warehouse that uses the dimensional model is made up of two types of tables: fact and dimensional. The first of these, the *fact table,* stores data (known as *measures*) representing quantitative information. The second, the *dimensional table* , appropriately enough, describes dimensions such as time or product groups. Each dimensional model has only one fact table but can have any number of dimensional tables. The primary key of the fact table is the concatenation of the foreign keys to all the (single) primary keys in the dimension tables. This type of relationship is known as a *star join* or a *star schema,* due to the fact that the arrangement lends itself to being drawn in the shape of a star, with the fact table at the center and dimension tables arranged around it at the points of the star. Columns within the dimension table are usually highly denormalized, with many of

the columns relying on each other. This type of arrangement allows each of these columns to be used as column headers in reports derived from the data warehouse.

Data within a data warehouse are commonly stored in a multidimensional structure known as a *cube* . A cube is a subset of the entire data warehouse and allows targeted analysis to performed on specific data. A cube is constructed by identifying the columns to be analyzed from the fact table and the dimensions of the analysis that should occur over from the dimension tables. By combining these dimensions, we have a cube comprising a number of cells that have a measure (or piece of data) from each of the intersecting dimensions.

Populating Data Warehouses

Before data warehouses can be used, they need to be *populated* with data. Data Transformation Services (DTS) is the tool used to achieve population. The process of populating a data warehouse has a number of steps you need to understand. The data bound for the data warehouse could well have come from a number of sources and is unlikely to be in a consistent structure. If these data are to be stored in and used from a single data warehouse, they must take on a common form. Furthermore, the data must be massaged so that they can assume this form. Consider, for example, a number of source databases capturing Boolean (TRUE or FALSE) responses; some might use TRUE and FALSE, others might use 0 and 1, others still might use Yes and No. A common form must be decided on, and all nonconforming data modified to fit this agreed standard. Similarly, you might need to ensure that all data fit within other constraints (for instance, particular ranges of values) and modify them to fit, based on some predefined logic, if a problem arises. Another common issue is the need to supply default values for fields that might have been left blank in source databases but are required to be not null in the data warehouse.

A number of options exist to carry out the transforming and cleaning of data. These tasks can be done by hand, which, although sure to be time-consuming and painful, is often the only option at some point, particularly if a diverse number of uncommon decisions need to be made. Alternatively, you can use the older data transfer mechanisms, such as bulk copy program (BCP). Transact-SQL commands can also be used to process data and create new, modified records based around logical decisions. However, DTS is a preferred option due its ability to create scripted packages. The process for using DTS to load data is described in the following section.

Using DTS to Transform and Load Data

Populating a data warehouse involves a number of steps. Often the data must be assembled from a number of sources, possibly even different databases from a range of database vendors, converted from existing schemas into a common intermediate schema, and cleansed to remove any inconsistencies, thus ensuring that the same types of data are unified into identical types and formats. Data are loaded into a data warehouse via a DTS task known as the Analysis Services Processing task. This task is accessed via the DTS

Designer and can be incorporated into a DTS package for execution immediately or at a later scheduled time.

In order to design a DTS task to load data into your data warehouse, follow these steps:

1. Launch Enterprise Manager and expand the tree of the server you want to work with.
2. Right-click the DTS folder, and select New Package from the pop-up menu.
3. Drag the Analysis Services Processing Task icon (the yellow cube) onto the design sheet (the blank space) to start the design process rolling. This action launches the Analysis Services Processing Task dialog box.
4. Enter a more appropriate name and description for the task, if you want.
5. Expand the server tree to enable you to select the object with which you want to work. In the screen in Figure 10.4 , you can see we are working with the Sales cube from the FoodMart sample.

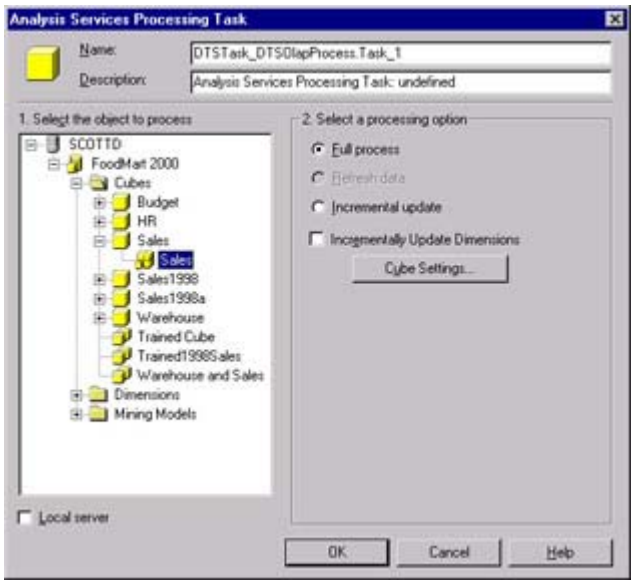


Figure 10.4: Working with the Sales cube in the Analysis Services Processing Task dialog box.

6. Determine the processing option you require. Different circumstances call for different choices from the three available processing options. When first loading data into a data warehouse, you must select the Full Process option. You must also select Full Process if the structure of the cube, or any of its dimensions or measures, changes at any time. Choosing Incremental Update is appropriate when new data need to be brought into a cube but the existing data and structure remain unchanged. The Refresh Data option should be used when the structure of a cube remains unchanged but its data need to be updated. For example, this method would be used when data in the underlying data warehouse changed. A fourth option, Incrementally Update the Dimensions of this Cube, should be used if any of the dimension tables have had new rows added since the last cube processing.

7. Click the OK button to finish creating the task, and then choose Save from the Package menu to save it in a DTS package. There are several formats in which a package can be saved; in this situation, choose to store the package in SQL Server, which causes it to be stored as an msdb table and offers you the best feature set when it comes to keeping track off and working with this and other packages.
8. Execute the DTS package now or schedule it to run later. If you choose to execute the package now, you can run it from the DTS Designer by right-clicking the package and selecting Execute from the pop-up menu. To schedule the package to run later, you first need to close the DTS Designer. Once you are back in Enterprise Manager, ensure that the tree is expanded so that you are able to see the your package in the right-hand pane. Right-click your package in this pane, and choose Schedule Package from the pop-up menu. This action launches a new dialog box; fill in the details as required to allow your package to run at the appropriate time. Packages can be scheduled to run either once or at recurring intervals of your choice.

Unattended Installations

At times, you might like to duplicate the options made in an earlier installation of Analysis Services or roll out the software without having to physically attend the installation. For example, you could have a server to set up at a remote location with only a low-bandwidth connection to your location, or maybe you need to roll out a number of servers across your organization and want to ensure consistency between the components installed on each machine. In this situation, an unattended installation is the answer.

In order to perform an unattended installation, you need to use the Analysis Services Setup program from the command line. The Setup program can be found on the SQL Server 2000 CD-ROM at Msolap\Install\Setup.exe. In order to prepare for future unattended installations, you need to record the steps taken in the desired installation process. You can do this by running the Setup program with the –r command-line parameter. This will create a silent response file in the systemroot directory (by default, C:\WINNT) with the .ISS extension. This file records the installation steps taken, particularly the input offered by the administrator, and can be used later to effectively take the role of the administrator during future installations.

When you want to undertake an unattended installation, you can use the (.ISS) response file, together with the media containing the Analysis Services Setup program and supporting files. When you run the Setup program with the –s parameter, the silent response file will be used in place of the administrator's input, in order to automate the process. By default, the Setup program searches for the response file in the same directory as itself. This might not be convenient, especially if CD-ROM media are being used. However, don't despair—the –F1 parameter can be used to define the location of the file. For example, the following command would perform an unattended installation using a silent response file (named Setup.iss) located at C:\Installs:

```
Setup.exe -s -f1C:\Installs\Setup.iss
```

Warning The act of processing a cube involves no checks on referential integrity. Take, for example, the situation in which one of the foreign keys from the fact table does not exist in the appropriate dimension table. In this case, no error will occur; instead, any rows with that particular value will not be processed, resulting in a functioning but incomplete cube that can no longer be relied on for analysis. As you might have inferred, the real danger here is that, because no indication of the problem has been given, the cube will still be used as though nothing is wrong. Therefore, it is critical that you verify integrity between the fact and dimension tables after the data warehouse has been populated with data.

Creating an Analysis Services Database

The process of creating an Analysis Services database has a number of steps. Broadly speaking, both the data preparation area and the database itself must be created before data can be loaded from other sources.

Create a Database and Data Source

The *data preparation area* is a relational database in which the data are prepared for transfer to the actual data warehouse database. This database can be a separate database or can be created within the same database as the data warehouse itself. Whichever approach you choose, you need to create a number of database objects (tables, scripts, and stored procedures) to carry out the extraction, cleaning and transforming, and eventual loading of the data into the tables of the data warehouse. At a minimum, the data preparation area should contain tables to store the data extracted from other sources and tables to store the transformed data, as well as the scripts or stored procedures used to extract and transform the data. The latter tables should have an identical schema to that of the actual data warehouse tables or, at a minimum, have a schema that allows the data within the tables to be transformed to the schema of the target tables within a single DTS process.

The physical process for creating the actual data warehouse tables will be familiar to you—it is the same as creating tables for an OLTP system and hence can be achieved using either T-SQL CREATE TABLE statements or Enterprise Manager. Basically, the task of creating the database is straightforward: We need only create the fact and dimension tables, create the relationships between them, and establish indices on the key fields in each of the tables.

To set up a new database, launch Analysis Services and right-click the server that will house the database. Select New Database from the pop-up menu, and enter a name and an optional description for the database. Once you have a database, you need to set up a data source to hold information about the OLE DB provider, security, and general connection

issues, such as which server is involved. In order to set up a new data source, you need only right-click the Data Sources subfolder of the database you just created and choose New Datasource from the pop-up menu. Fill in the options from the Data Link Properties dialog box that is displayed, selecting the OLE DB provider to use (in this case, select Microsoft OLE DB Provider for ODBC Drivers), and from the Connection tab, specify the data you want to use. Once this is done, you are ready to take the next step in creating your OLAP solution: designing and building the cubes to analyze.

Designing and Building Cubes

A cube is a (multidimensional) structure that contains data from a data warehouse. A cube can contain the full set of data from the data warehouse; commonly, it contains a subset of the data. Hence a dimensional database can contain a number of cubes, each one built to allow querying of different sets of data. Although it is possible to directly query a dimensional database, this is a time-consuming process because transactional data need to be aggregated at query time. The use of cubes allows much faster responses to such queries.

The creation of a cube contains a number of steps you can perform using the Cube Editor or the more user-friendly Cube Wizard. We use the Cube Wizard in the following section to put together our own cube to query the FoodMart2000 database.

Using the Cube Wizard

From the Analysis Manager application, expand the FoodMart2000 tree until you can see the Cubes folder. Right-click this folder, and choose New Cube from the displayed pop-up menu. Select Wizard from the two choices offered in the fold-out menu. Skip past the Welcome screen until you get to the next screen, which asks you to select a fact table. If you were involved in the design of the dimensional database, you already know which fact tables are available, so this should be a simple step. However, at times (like this one) you might be working with a database designed by someone else. In these cases, be aware that although the wizard displays all tables, not all of them are suitable for use as fact tables; only those with numeric columns (measures) can be used.

As an example, let's build a cube based on the sales_fact_1998 fact table. Follow these steps.

1. Highlight this table, as shown in Figure 10.5 , and press the Next button to progress through the wizard to the next stage. The columns in the details section are shown for reference only; there is no need to attempt to highlight or use them at this time.

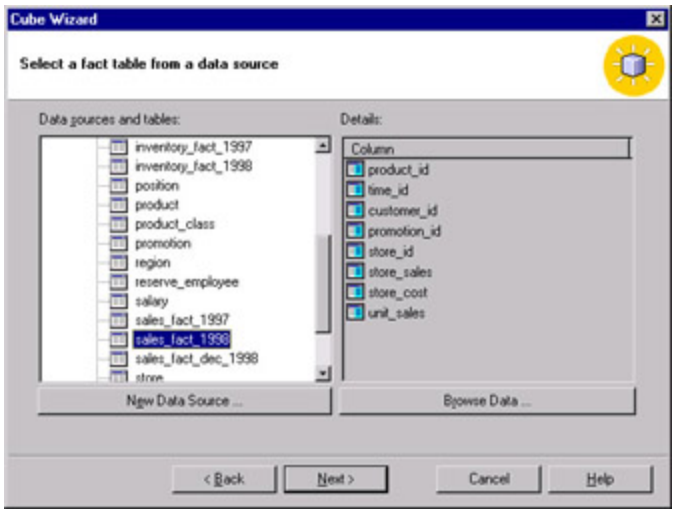


Figure 10.5: Selecting a fact table with the Cube Wizard.

2. The next step requires you to select the measures you want to establish within the cube. Remember that measures are the items that you would like totaled across the dimensions, so here choose the items for which you would like to see summary data. In our example, we will monitor Store Sales and Store Cost. To select these items, highlight them from the list and move them across to the right-hand (cube measures) pane with the top-most single arrow. Figure 10.6 shows you what the wizard should display once this is done. Press the Next button to move along to the next stage.

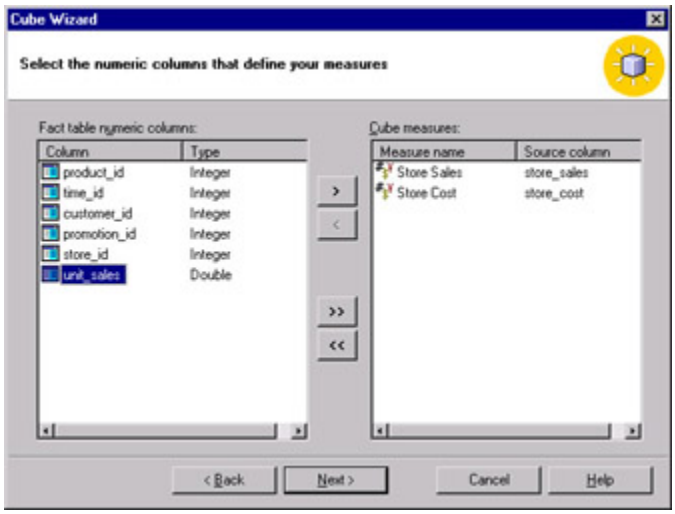


Figure 10.6: Selecting measures with the Cube Wizard.

Tip Measures can be either made up of a single column, as in our example, or made up of a number of columns related by an expression. In our example, we could create a Gross Profit combined measure that would be Store Sales less Store Cost. The Cube Editor is the tool to use to create combined measures.

3. The next step in the wizard asks you to select the dimensions for your cube. These are the dimensions across which you want to analyze the data. For example, you might want to look at your data across certain time periods or examine sales across specific geographic areas. In our example, we'll analyze our data across stores, store types, store sizes, and time, so we will choose these items as dimensions for this cube. Again, highlight the items in the left pane and move them across using the topmost single arrow. Figure 10.7 shows how your screen should look once this is done.

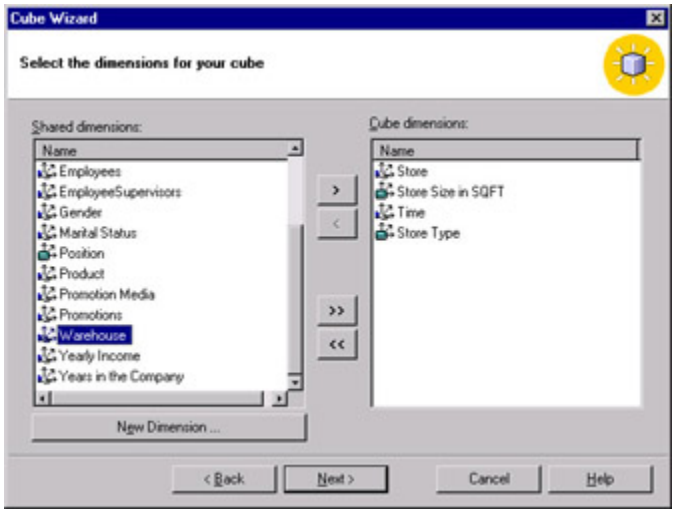


Figure 10.7: Selecting dimensions with the Cube Wizard.

4. After selecting the Next button, you will be warned that the process could take a while; accept the warning and let Analysis Services create the cube for you. Once the process is done, you will be shown a tree view of the cube structure, which you can browse, and you will be asked to name the cube. Enter the name **Sales1998** and press the Finish button. Figure 10.8 shows the results of your work, as displayed in the Cube Editor, which will now open automatically.

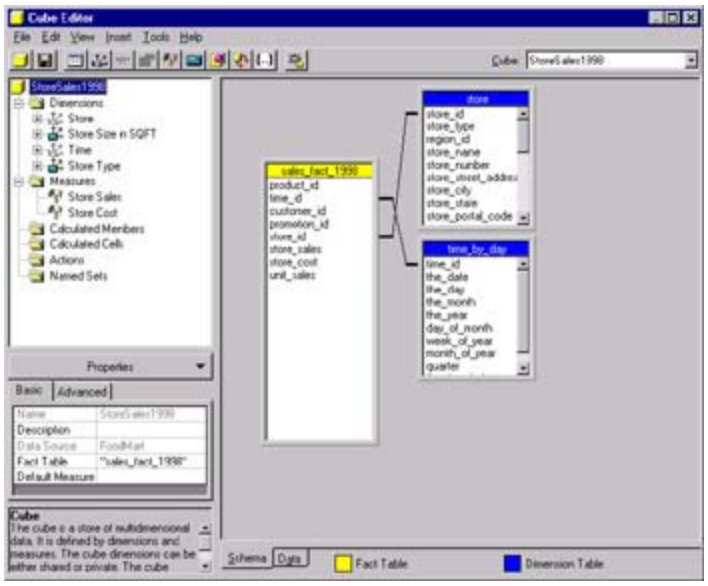


Figure 10.8: The results of the Wizard: The Cube Editor.

5. Close the Cube Editor, and answer Yes when asked if you want to save the cube and design data storage options for it.
6. The Storage Design Wizard will launch; again, skip past the Welcome screen.
7. Now choose a storage option from the three choices presented. For our example, we'll use multidimensional OLAP (MOLAP) because it will store the data on the Analysis Server in a proprietary compressed format and will provide the quickest response to queries of the three types.
8. Once you have chosen a storage option, you will be asked to make a choice with regard to aggregation. *Aggregation* is basically the summarizing of data and the storage of those results in advance for use in later queries. The level of aggregation determines both the amount of disk space required to store the cube and the speed with which results can be returned when queries are executed against the cube. Two extreme examples illustrate this point. If we use no aggregation on our cube and the user wants to see sales for the last quarter across all stores, the database has a mammoth task to complete. Even if there are only 100 transactions per day per store and 50 stores across the company, the query processor must read and sum 100 x 50 x 30 x 3, or 450,000, transactions to come up with the result. If instead we aggregate the sales per store per month, we have only 50 x 3, or 150, rows to process. On the other hand, if we were to aggregate data to attempt to handle all possible scenarios and combinations, then although the theoretical performance would be impressive, the amount of storage space required to achieve it could not be justified.
9. You can choose to control the level of aggregation based on the amount of disk space consumed by the cube and the likely performance gains or you could let the wizard run until you click Stop. In our example, choose this last option, and press the Start button. Accept the suggested number of aggregations (likely to be about 12 in this example), and press the Next button to continue.
10. Choose to process the cube now, and press the Finish button to prepare the cube for use.

11. Once processing is complete, the cube is ready for use. You can browse the data by right-clicking the cube and selecting Browse Data from the pop-up menu.

Note Choose MOLAP for cubes that will be processed often and for which speed is the most important factor. Select ROLAP for large databases in which data are infrequently updated. ROLAP provides slower response times because the base data are left in a relational form. Due to this structure, views can be used to provide data summaries when ROLAP storage is used. *Hybrid OLAP (HOLAP)* is a mixture of MOLAP and ROLAP in which aggregated data are stored in a multidimensional structure but the base data remain in a relational structure. HOLAP cubes are faster at retrieving summary data but slower for drilling down to retrieve individual data items; for that reason, they are best used for summary information based on a large amount of base data.

Editing Cubes

Once built, cubes can be edited using the Cube Editor. This is accessed by right clicking on the cube and selecting Edit from the pop-up menu. Once inside the Cube Editor a number of cube properties may be viewed, delete and / or changed via a right click. These include dimensions, measures and calculated members. Editing of other cube settings such as aggregation and storage design is available from a right click on the cube in Analysis Manager and selecting the appropriate entry from the pop-up menu. Once selected these items are edited using the now familiar tools and interfaces we used when building our example cube.

Data Security in Cubes

Access to cubes by users is determined via cube roles. Users only have access to cubes that have been assigned a cube role containing that particular end user's name. The defaults for a cube role are provided by the database role of the same name.

Database Roles

Following the creation of a database role it can be assigned to any cube using Cube Role Manager, which can be accessed by right clicking the Database Roles node of the tree within Analysis Manager. Via this interface access to cubes can be granted to roles using the Cubes and Mining Models column, and database users can be added or removed from the role by using the Membership column. Although, not enabled by default it is also possible to set up dimension level security or cell level security which restricted the dimensions or cells of the cube that the user is able to see, or in the case or read—write restrictions only, update.

Defining Measures and Dimensions

We now have a cube but as yet it is little more than a shell. In order to start to use our cube for analysis we need to understand its internals and structure those to our needs. Measures and dimensions are those internals; let's look more closely at them now.

Understanding Measures and Dimensions

A measure is an item that can be quantified. It is numeric, and in OLAP it is this item that will be summarized and reported upon across various periods or areas—otherwise known as dimensions. Example measures include sales, profit and cost.

A dimension is a set of related attributes used to describe the measures from the fact table. Typical dimensions include time, product range and sales territories.

A member is the term used to describe each discrete component within a dimension. For example the time dimension has the members day, week, and month. Analysis of measures can occur across dimensions by any of the member breakups.

Creating and Editing Measures and Dimensions

Within Analysis Services, two types of dimensions exist: shared and private dimensions. *Shared dimensions* are dimensions that are shared among more than one cube within the data warehouse. The most obvious of these dimensions is that of time, which is used to some degree in nearly all OLAP solutions. You might also hear shared dimensions referred to as *conformed dimensions* . *Private dimensions* are created for and are able to be used within individual cubes only. Shared dimensions are preferred over private dimensions because the former save on storage space, introduce common business metrics where possible, and negate the need to rebuild the cube whenever a dimension is changed (as it the case with private dimensions). For these reasons, we concentrate our discussion on creating a shared dimension.

Before we can create a shared dimension, we must consider a number of other items. The data source containing the tables and columns that define the dimension must be known, and the levels that shared dimension will contain must be determined. Once these items have been agreed on, the process of creating the dimension can begin.

Two tools exist for the creation of shared dimensions: the Dimension Editor and the Dimension Wizard. We look at the wizard in this example and then look at the editor when we examine how to edit a dimension. Follow these steps to use the wizard:

1. Right-click the Shared Dimensions folder in Analysis Manager, and choose New Dimension and then Wizard from the resulting menus.
2. Now choose the type of dimension to be created from the choices presented. In our example, we will choose Parent-Child. The uses of each type are as follows:

- **Star schema** Creates a regular dimension, such as product, from a single dimension table.
- **Snowflake schema** Similar to the star schema, but this option deals with multiple, related dimension tables. Recall our discussion of star and snowflake schema at the beginning of this chapter.
- **Parent-Child** Creates a parent-child dimension from two related columns in the same dimension table. For example, an e-mail message and its reply would create a message thread dimension.
- **Virtual** This type of dimension is not calculated until runtime and hence uses no disk space.

3. On the next screen of the wizard, select the table within which the dimension columns exist. For our example, we will use the Employee table; highlight it and press the Next button.
4. We now need to define the relationship in the parent-child hierarchy. We will set up a relationship between an employee and her supervisor. To do so, select EmployeeID as the member key and SupervisorID as the parent key. (You will see a self-join appears on the table displayed in the wizard.)
5. Now select FullName as the member name to give an intuitive means for users to browse the dimension. Figure 10.9 shows how this screen should look once you've completed this step. Press the Next button, and move to deal with the Advanced Options.

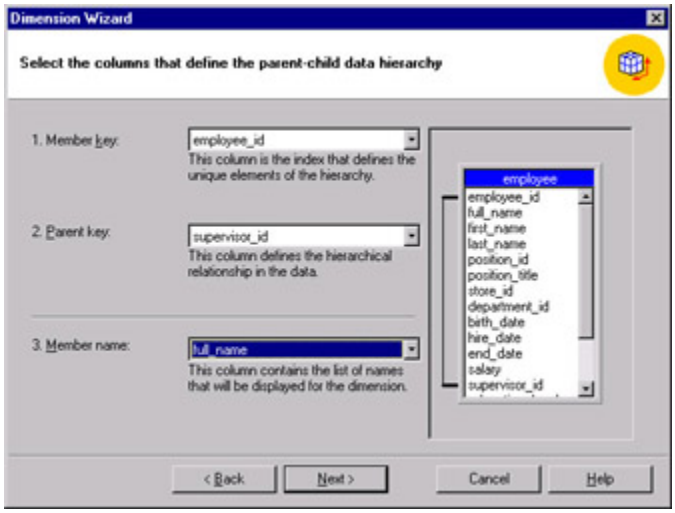


Figure 10.9: Defining the parent-child hierarchy in the Dimension Wizard.

6. Of the features here, the ones of interest are the Members With Data option and the settings associated with it on the next screen, namely Non Leaf Members Have Data and Data Is Visible. This group of features allows the data associated with a member of a dimension to be rolled up into that hierarchy during consolidation while still being recognized as data having originated from a specific member. For example, consider a sales manager with sales reps making sales in the field. The manager also makes sales, but unless this option is used, the manager's sales will not be counted correctly in a hierarchical roll-up. Check the Members With Data option, press the Next button, then select the Non Leaf

Members Have Data option and click the Data Members Are Visible radio button before moving on.

7. To finish the process, give the dimension a name and browse the data within it.

Once the wizard has completed its task, your new shared dimension will be displayed in the Dimension Editor. This is the tool of choice for making changes to existing dimensions. It can also be accessed by right-clicking the dimension in question from Analysis Manager and choosing Edit from the pop-up menu. Once inside the editor, you can insert, modify, and delete items such as tables, joins, and member properties.

Using Your OLAP Solution

Now that you've built a basic OLAP solution, what next? The data are in the data warehouse, the cube is built and in place, but we have yet to see any results from our hard work; to date we have no output. In the following section , we look at how to use our OLAP solution.

Querying Cubes

In order to extract information from our data warehouse, we need to query our cube. Querying cubes is the means by which the historical data in the warehouse can be examined to return useful information about any number of questions. If you're familiar with SQL or you've read this book from the beginning, you know that you can query database tables using SQL. You can also use a number of means to query the date warehouse data source (the cube), although the process is different from querying database tables. Let's look at the options.

HTTP Cube Access

SQL Server 2000 introduced the concept of being able to access database objects using Hypertext Transport Protocol (HTTP). Put simply, this feature allows access to the SQL Server 2000 database via the public Internet and private intranets. At this point, just remember that queries can now be passed to SQL Server via a URL and the results returned via eXtensible Markup Language (XML) for display in a Web browser and that when you access Analysis Services via HTTP, you must use the PivotTableService, with the data source set to the URL of a computer running IIS that has access to the SQL Server. Once the system is authenticated and connected, communication is seamless to the client application, and hence access techniques used across a LAN or WAN are also applicable when connecting across the Internet.

Analysis Server can be queried across either HTTP or the secure HTTPS of Secure Sockets Layer (SSL). The process in the two cases is quite similar. For example, consider that you need to access a cube you have created while working your way through this chapter from an ASP page. Use the following code when you work with HTTP to access data in a virtual directory named olap_location on your Web site named mysite:

```
Dim objCatalog as new ODBMD.Catalog
cat.ActiveConnection = "Provider = msolap;" & _ " Datasource =" _ & "
http://mysite/olap_location;" & _ " Initial Catalog = FoodMart 2000"
```

To access the same solution across SSL, replace *http* with *https* .

Multidimensional Expressions

MDX is a syntactical means of defining and querying multidimensional objects from data warehouses. Although MDX might appear to be similar to SQL, it is not an extension of

it. Although MDX does support a set of Data Definition Language (DDL) commands for creating, modifying, and otherwise manipulating cubes, we will not touch on those aspects here.

MDX functions fall into a number of categories, each of which provides different capabilities. The following is a list of the types of categories, grouped by return types:

- Array functions
- Dimension functions
- Hierarchy functions
- Logical functions
- Member functions
- Numerical functions
- Set functions
- String functions
- Tuple functions
- Miscellaneous functions

Instead, we concentrate on using MDX to query a cubes and retrieve information. MDX statements have three primary constructs:

- A request for data (SELECT statement)
- A starting point (FROM)
- An optional filter to restrict the amount of data returned (WHERE clause)

As with SQL, these constructs allow strings of commands to be built to return sets of data. We'll ease our way into it in order to build a good firm basis to develop an understanding from. Analysis Services ships with a sample MDX application that can be used to build and execute MDX expressions against the database. The basic syntax of an MDX expression is as follows:

```
SELECT axis specification ON COLUMNS,
       axis specification ON ROWS
FROM cube_name
WHERE slicer_specification
```

Think of the axis specification as a dimension; more axis specifications suggests more dimensions. The WHERE clause is optional, just as it is in SQL.

Let's look at a basic MDX statement to query the Sales1998 cube we built earlier. This query will return the sales for stores from the United States and Canada from each of the first two quarters of the year. Use the MDX Builder to enter the following query. You'll find the builder in the Analysis Services program group:

```
SELECT ([Canada], [USA]) ON COLUMNS,
([1998].[Q1], [1998].[Q2]) ON ROWS
FROM SALES1998
```

This is a basic but nonetheless powerful query on which you can expand and use to model MDX queries on other cubes.

MDX expressions can also contain *calculated members,* which are values derived from other measures in the cube. If you look back at our Sales1998 cube, you'll see that we track both sales across time. We might want to see sales growth from quarter to quarter. The following MDX query will show growth between quarter one and quarter two of 1998:

```
WITH MEMBER [Time].[1998].[SalesGrowth] AS '[1998].[Q2]—[1998].[Q2]'
SELECT ([Canada], [USA]) ON COLUMNS,
[SalesGrowth] ON ROWS
FROM SALES1998
WHERE [StoreSales]
```

This query is slightly more complex than the first one, so let's go over it together and see what the new sections achieve. Our calculated member is SalesGrowth and is defined by the WITH MEMBER statement. We prefix it with [Time].[1998] because the levels of the hierarchy must be maintained. We use the AS keyword to allow the definition of the formula that will construct the calculated member. We are now selecting the U.S. and Canadian stores as one axis and the quarter-to-quarter sales growth on the other axis. The WHERE clause (or the *slicer specification,* as it is more correctly known) is used to outline the slice of the cube to be viewed. This equates to defining the measure to be reported on, in this case the sales that are held in a member named StoreSales.

MDX expressions can be extended and developed into very complex structures. They can be used to query and even create multidimensional structures. Much time could be dedicated to exploring the syntax more deeply and explaining many of the other aspects of MDX, such as named sets, but that is left as an exercise for you, because the intent of this section was merely to introduce you to the concept of MDX and show you how it can be used to query multidimensional data in cubes.

Data Mining in SQL Server

Data mining is used to unearth patterns in data. These patterns can appear obvious in hindsight, but more often they are subtle trends that would go unnoticed without tools such as those provided by Analysis Services. Data mining is a highly automated process; it needs to be so, due to the high volume of data that needs to be trawled through in order to search for and find patterns. The second task of data mining is to use the discovered trends to predict future patterns and behaviors for new areas based on those patterns. In a nutshell, a data-mining tool moves through data, seeks relevant information, and presents that information to you.

Mining Models

Broadly speaking, a *mining model* gives the definition of and boundaries for the data being mined and the type of predictions you are looking to make from the mining process. Analysis Services is able to mine both relational and dimensional databases.

Relational Data-Mining Models

A *relational data-mining model* is used for mining data from a relational database. This mining model is best used when working with rapidly changing data. In such environments, it might not be feasible to create a dimensional database from the relational model on a regular basis. Think of a call center operation in which mining of likely response times and key performance indicators is required on a regular basis; this is a good candidate for using a relational data-mining model.

OLAP Data-Mining Models

An *OLAP data-mining model* is used for mining data from a dimensional database. This is the model you will work with if you are mining a cube, as we will in our example. These types of mining models are best used when the data being mined are rarely changing. Consider mining sales information for a chain of stores. The data are brought into the (dimensional) data warehouse at the conclusion of each day's sales, when historical data are available for mining. The mining model and the data underlying it are not up to the minute, but because sales trends are longer-term concerns, this is not an issue.

Data Mining Algorithms

Data-mining algorithms are at the heart of the data-mining process. These algorithms determine how cases are processed and hence provide the decision-making capabilities needed to classify, segment, associate, and analyze data for processing. Currently, Analysis Services supports two algorithms: clustering and Microsoft decision trees. It also provides support for the OLE DB for Data Mining API, which allows third-party providers of data-mining algorithms to integrate their products with Analysis Services, thereby further expanding its capabilities and reach. The architecture is extendible; data-

mining algorithm providers are simply OLE DB providers, so it is likely that we will see more algorithm providers appear in the future, either supplied by Microsoft or by third-party vendors.

Microsoft Decision Trees

Decision trees arrange information in a tree-like structure, classifying the information along various branches. Each branch represents an alternative route, a question. This structure can be used to help you predict likely values of data attributes.

Microsoft Clustering

Clustering groups like data together in various groups. There is no predetermined arrangement for grouping the data; similar data are assimilated, and the analysis of the significance of such groupings is left to the user. Clustering is useful when you are trying to see patterns in data, such as trying to identify geographic regions that are likely to respond well to a certain sales campaign.

Creating and Editing Data-Mining Models

One of the early stages of the data-mining process is to develop the data-mining models. This is, in reality, a group of processes encompassing the design and setup of the model, training the model with existing data, and editing the model in order to refine it to ensure that predictions made for it provide as much value as possible. These processes are described in the sections that follow.

Mining Model Wizard

A new mining model can be created with the Mining Model Wizard. To launch the wizard, right-click the Mining Models folder from inside Analysis Server, and select New Mining Model from the pop-up menu. Your first choice is to determine whether you are mining relational or dimensional data. In this case, we are mining data in a cube, so select the OLAP Data option, and press the Next button. Had we chosen to mine relational data, we would see slightly different options as we moved through the wizard. The concept however, remains the same.

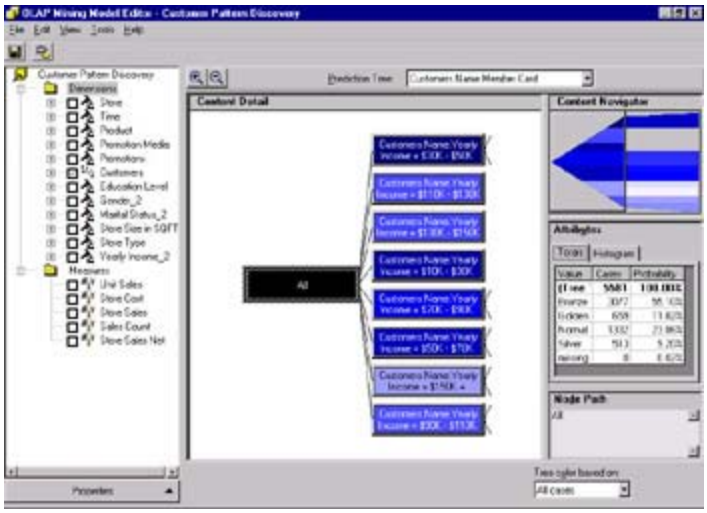
You now must select the cube you want to mine; for this example, select the Sales1998 cube you created earlier. Now you must select the mining technique to use in this instance. This equates to the algorithm to use. Analysis Services currently supports two techniques: Microsoft Decision Trees and Clustering. Your choice here will depend mainly on how you want to use the results. Using a decision tree will prove of more use when you need to make specific predictions from the model, such as whether or not a specific customer is likely to be able to repay a loan, whereas clustering is good if a general grouping is required, such as indicating which groups within a geographic area are most likely to buy a particular product based on various demographic indicators. In this case, we will use Microsoft Decision Trees, so select it now and press the Next button to move on.

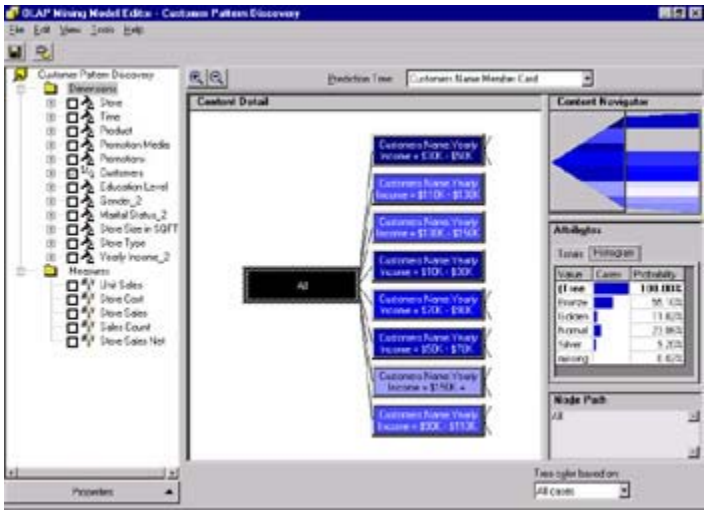
The next step involves choosing a data-mining case, which equates to the entity from the data that you intend to mine. In our example, we are interested in analyzing data about stores, so select Store for the dimension and Store Name for the dimension level. Next you must determine the data you want to predict. In our case, we are concerned with predicting store sales, so select that value and move on. The next stage of the wizard involves selecting the data that will be used to train the model before it is able to make predictions. The default values will most often be fine to use. Notice that in our example, Store Sales is listed among the training data, essentially making it an input as well as an output. Without this step, the model would have no idea what effect other items would have on sales and hence would be useless as a prediction tool.

Allow Analysis Services to create a new virtual cube based on the Sales1998 cube and a new data-mined dimension by given them names in the spaces provided. Although the names are up to you and don't affect performance, it's a good idea to use the prefix Predict for the new dimension (for example, Predict1998Sales) and the prefix Trained for the virtual cube (for example, Trained1998Sales). This type of descriptive name helps identify at a glance what each of these objects is and does. Give the model a name, and select the Save and Process Now option, which will create and train the model. This process will be quite quick for our example, but it is likely to take quite some time for larger, real-world examples. When processing is finished, the OLAP Mining Model Editor is displayed.

OLAP Mining Model Editor

The OLAP Mining Model Editor is used to display the results once processing and training of a mining model is complete. A number of panes appear on the screen, as shown in Figures 10.10 and 10.11 . The large middle pane shows a number of nodes from the decision tree; the upper-right pane models the whole tree, using colors to represent data density. When you select one of the nodes, the attributes section (the bottom-right pane) changes to reflect the node currently selected. The attributes show the breakdown of the entities within the group that the node represents, which can be shown numerically as in Figure 10.10 or graphically with a histogram, as in Figure 10.11 . Navigation throughout the model is possible either using the Content Navigator at the top right of the screen or by double-clicking a node to drill through it.


Figure 10.10: Results obtained using the OLAP Mining Model Editor, showing numerical attributes.


Figure 10.11: The browser showing attributes graphically.

Using Data-Mining Models

Once the data-mining model is established, you can use it to make predictions. In this section we look at how models can be used and the steps required to ensure that the predictions obtained are less likely to be in error.

Data-Mining Training

Before a data-mining model is able to produce predictions of any worth, it must go through a period of training. During this training period, the model works with a set of known (actual) data as it attempts to identify patterns, rules, and trends among that data. The concepts behind the training process are similar across OLAP and relational mining models. Put simply, the training data are expected to be found in the source tables or cubes used to build the mining model.

Training the model occurs by processing the model from either the Mining Model Wizard or from one of the two mining model editors. There are two ways that a mining model can be processed. The *full-process option* first removes and then completely rebuilds the mining model and trains it from scratch with the training data; this is the method of choice when a new model is constructed or when the structure of a model has changed. The alternative is to *refresh* the mining model, which leaves the structure intact but clears the model and retrains it from the training data. This is the preferred choice when it is thought that trends might have changed and a fresh set of training data is required to ensure that accurate predictions are being made from the model.

Note Users can continue to access a data-mining model during and after a refresh takes place. However, when the full-process option is used, users must disconnect from and reconnect to the server before they can work with the modified model.

Data-Mining Model Browser

The *data-mining model browser* is a highly visual tool. It provides a means of easily identifying patterns within the data. It does so by providing a number of panes, thus allowing a high-level, color-coded overview to show patterns and clustering, as well as providing both graphical and statistical information about each of the nodes in the browser. It is also possible to drill through from nodes to the data that lie below and then gain the same sort of detail about that data. Figures 10.12 and 10.13 show two different views within the mining model browser and illustrate each of the panes that are used to display various aspects of the data.

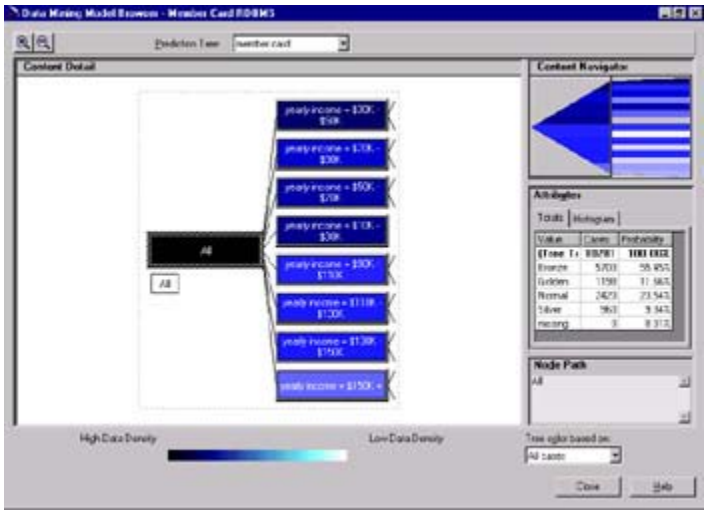
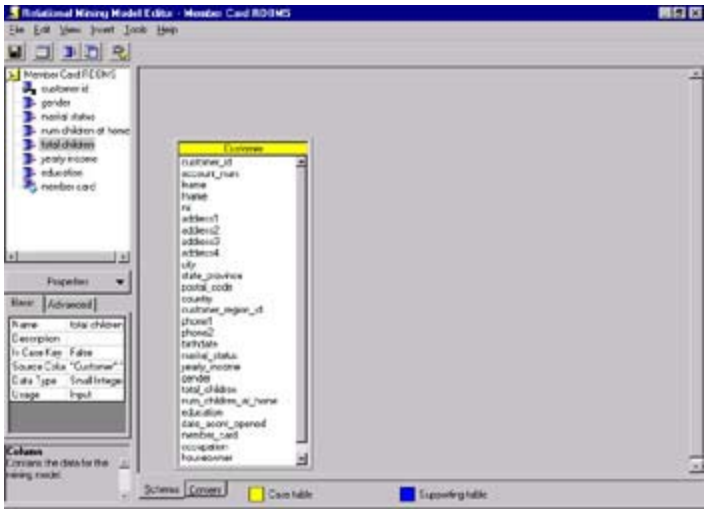


Figure 10.12: The mining browser, showing results from the Relational Mining Model Editor.


Figure 10.13: The Relational Mining Model Editor.

Discovering Patterns in Your Data

The highly visual nature of the browser adds a better understanding of the trends and patterns within the data. It also makes it easier to fine-tune the model to better suit the overall trends rather than *over-fitting,* or over- emphasizing, specific trends that might be seen in the data when a narrower focus is used. Over-fitting can lead to less effective predictions from the model because all trends are not accurately reflected.

Let's look at few items that will help you understand and read the view in front of you:

- The *nodes* (represented by the colored shapes in the model browser) are color-coded to represent the data density of an attribute applicable to a selected node and in relation to the total number of cases processed by that node.
- The nodes are represented in ranking order of attributes. The further to the right in the tree a node is located, the less influence the attribute has within the data-mining model and the less likely the decision represented by the node will have a substantial effect on the predictions made by the model.
- The attributes pane allows ranking of the attributes by number of cases or probability of occurrence in the selected node; this facilitates a better understanding of the importance and relevance of a given attribute to a node.

Multidimensional Expressions for Data Mining

MDX expressions can be used as alternatives to using the data-mining model browser to look at patterns in your data. These expressions could be written and used in a scenario in which users lacked the OLAP administrator rights to run Analysis Manager, or there could simply be a need to get this type of data into another application for further analysis or custom processing of some type. In order to use MDX in this fashion, you need to use OLE DB for Data Mining.

ActiveX Data Objects Multidimensional (ADOMD) provides a library for accessing dimensional data from client applications, allowing the execution of MDX expressions against Analysis Services structures.

OLE DB for Data Mining

OLE DB for Data Mining is an interface that allows third-party clients to be used to perform data-mining operations against SQL Server databases. This essentially means that developers can implement solutions that not only use traditional OLTP systems but now can also provide data-mining functionality. Specifically, the use of the OLE DB for Data Mining interface allows the client application to connect to an Analysis Server, browse a multidimensional schema, execute queries against cubes, and bring the results back to the client.

Security in Analysis Services

Analysis Services provides several alternatives for restricting access to the data it contains. These options are discussed in the following section .

Users and Groups

Before roles can be created, you must set up one or more users and groups. Setup of these users and groups is not done from SQL Server but rather from the User Management tool of the specific operating system in use. If you are using Windows NT 4.0, your tool of choice is User Manager for Domains, which is located in the Administrative Tools program group. If your OS is Windows 2000, you need to use Computer Manager, which is accessible via Administrative Tools in the Control Panel. Try to finalize group membership here before moving on to create roles; it will make role maintenance easier.

Roles

A (security) *role* is a grouping of Windows NT or 2000 users or groups with the same level of access to Analysis Services data. Analysis Services has three types of role:

- **Database role** This role is assignable across multiple cubes or mining models and grants read-only access by default. A database role provides the defaults for a cube of the same name.
- **Cube role** This role applies to a single cube only and inherits its defaults from the database role above it. However, tighter control can be enforced at this level by overriding some of these settings, applying further security at a cell level, or preventing drill-through on the cube.
- **Mining model role** This type of role applies to a single mining model and, like the cube model, gets its default settings from the database role of the same name. Defaults can be overridden at this level.

It is possible for a user to be in more than one role. In this scenario the user has access to an object if any of the roles provide access to that object, regardless of whether or not other roles block that access. One possible exception to this rule is the situation of combinations of custom rules from dimension security; not all combinations of these roles are able to be resolved successfully.

Note All of the setup and allocation of users to roles could be in vain if the user is unable to be successfully authenticated when he or she first connects to the Analysis Server. Without successful authentication, no data can be accessed, regardless of the access levels defined in the roles in which the user has membership.

Data Security

Data security is provided within cubes by roles. Unless a cube is assigned a role that has the user's name as a member, no access can be granted that user. Assuming access is given, the level of that access depends on the particular access levels set up in that role. By default, the user has only read access to data, but if the role has been write enabled, the user can update data as well.

Further levels of fine control can be gained by setting dimension security, which restricts the dimensions users can see as they browse a cube. It is not mandatory to set up dimension security, because by default, the role has access to allow dimensions within a cube. Security for shared dimensions is done at a database level; other dimensions have security set at the cube level.

It is also possible to implement cell security within cubes. Again, this is optional, with users having access to all cells within a cube by default. If you do choose to implement cube security, it is possible to stop users accessing particular cells within a cube and controlling their ability to write, rather than just read, certain cells.

Implementing Security in Analysis Services

Implementing security inside Analysis Services can be done at a number of levels. Here we discuss the options and cover the steps involved.

Once you have created Windows users, you are able to create database roles as follows:

1. Start Analysis Manager, and right-click the database in question.
2. Choose Manage Roles from the pop-up menu.
3. Choose New, and then choose the Database Role from the fold-out menus.
4. Enter a name and an optional description in the Role dialog box.
5. In the Enforce On box, select Client for better performance or Server for better security.
6. On the Membership tab, add the required users by pressing the Add button and double-clicking the groups required from the list provided. If you want to add individual users, click the Show Users button to add them to the list from which you select users.
7. Repeat the process for other users, and then click the OK button when you have finished.

The process of setting up cube or mining model roles is similar but is started from the relevant section of Analysis Manager—for example, by right-clicking the appropriate cube or mining model in the tree view and choosing New | Cube Role or Mining Model Role.

Accessing Analysis Services Over the Web

From SQL Server 2000, it has become possible to access the SQL Server database across the World Wide Web (WWW, or the Web). It is now possible to obtain access via a Web address, commonly known as a *Uniform Resource Location (URL),* that specifies HTTP as the protocol. SQL statements or, in the case of Analysis Services, MDX statements, can be set at the URL to execute against SQL Server. However, before it is possible to take this course of action, some configuration tasks need to be completed.

Configuring IIS for Analysis Services

Microsoft's Web server, known as *Internet Information Server (IIS),* must be installed in order to allow access to Analysis Services across the Web. IIS version 4 or later is required to be installed. In order to install it, the underlying OS must be either Windows NT Server 4.0 or Windows 2000. Note that for Windows 2000 Professional installations, the Administrative Tools pack must also be installed.

Follow these steps to use IIS with Analysis Services:

1. Copy the Msolap.asp file from the Analysis Services Bin directory (normally C:\Program Files\Microsoft Analysis Services\Bin folder) to the C:\Inetpub\Wwwroot folder on the computer running Analysis Server.
2. Start the Internet Services Manager (from Administrative Tools).
3. Expand the tree so that you can see the default Web site, and select it.
4. In the right-hand pane, right-click Msolap.asp and select the Properties option from the pop-up menu that appears.
5. Within the Properties box, ensure that the Read check box is selected, and then select the Script source access box. Click OK to confirm these properties, and exit the dialog box.
6. Bring up the Properties box of the Web site by right-clicking the site entry in the left-hand pane and choosing Properties.
7. On the Home Directory tab of the Properties box, make one of the following two choices. The choice you make will depend on the OS your computer is running:
    - For Windows 2000, select either Scripts only or Scripts and Executables from the Execute Permissions box.
    - For NT 4.0, select either Script or Execute (including script) from the same area.
8. Click OK to complete the process.

Performance Tuning and Optimization

Data mining and OLAP tasks are by nature very intensive operations. Performance tuning and optimization can be critical to organizations using these technologies, particularly those with very large databases. Often, optimizing database performance has been a hit-and-miss affair for many DBAs, who make changes and modifications based on trial and error and repeating operations that have had success in the past. Although this method can provide some performance increases, those increases are unlikely to be optimal.

Analysis Services provides a number of tools to assist in locating and rectifying performance issues. Among these tools and techniques are the careful structuring or restructuring of schema design and the use of performance counters with either Windows NT 4.0's Performance Monitor or Windows 2000's Performance tool. These techniques can be quite involved and time consuming. In order to get you up and running quickly, we'll look at two wizard-based solutions for tuning and optimization: the Usage Analysis Wizard and the Usage-Based Optimization Wizard.

A number of performance counters also exist for monitoring the performance of Analysis Services. Among these counters are objects that provide statistical information about queries, the performance of indexes in cubes, and connections to Analysis Services. Books Online provides a complete list of the numerous performance counters that are available.

Before any of these counters can be used, a number of entries need to be made in the registry to provide information about various components used by the counters. This task could be done manually, but it is achieved more easily by executing the following command, which makes the entries for you:

```
LODCTR msmdctr.ini
```

The LODCTR.exe application can be found on either the Windows NT 4 or 2000 CD-ROM.

Usage Analysis Wizard

The Usage Analysis Wizard is designed to give you an on-screen representation of the performance of queries executed against a given cube. It does so by providing a number of predesigned reports that draw their data from the query log. Briefly, the reports available and what they show are as follows:

- **Query Run Time** The runtime for each query, in descending order.
- **Query Frequency** The number of times each query has been executed, in descending order.
- **Active Users** A list of users that shows the number of queries they have sent, in descending order.
- **Query Response Graph** The response time for all queries.

- **Query By Hour Graph** The total number of queries processed each hour.
- **Query By Date Graph** The total number of queries processed by day.

Analysis of these reports can help you pinpoint potential bottlenecks in your OLAP solution.

Start the Usage Analysis Wizard from Analysis Manager by right-clicking the cubes you want to work with and selecting Usage Analysis from the pop-up menu. You then are able to select the type of report you want to use before being asked to apply one or more filters to the underlying data. These filters assist you in drilling down to the root of the problem more quickly, because they help eliminate the "noise" (ordinary data) that surrounds the bottlenecks. Often, you receive have reports from users about specific problems that occurred at certain times; use these reports as the basis for constructing worthwhile filters. Once the filters are in place, click the Next button to see the report presented on screen.

Tip The Usage Analysis Wizard actually reads only every 10th query from the query log. You can change the frequency of these reads if you desire by adjusting the value found on the Logging tab of SQL Server's Properties dialog box.

Usage-Based Optimization Wizard

Once you have identified specific queries that could be causing bottlenecks, you'll no doubt want to rectify the situation. The Usage-Based Optimization Wizard is the tool you'll use to do so. This wizard allows you to optimize the aggregation design based on past queries, hence achieving performance improvements on your cubes.

Follow these steps to use the wizard:

1. To launch the wizard from Analysis Manager, right-click the cube you want to work with, and select Usage-Based Optimization from the resulting menu.
2. If you haven't disabled the Welcome screen, skip past it and you'll see a filter screen very similar to that presented in the Usage Analysis Wizard. Look closely and you'll see an additional option at the bottom right of the screen: the ability to specify the type of storage the queries run against, whether MOLAP, ROLAP, or the server cache.
3. Once you have set the filter options, press Next to see a list of queries matching the specified criteria.
4. If you decide to make changes to the options or add aggregations, follow the steps of the wizard to select the type of storage you would like to use, and then design, save, and process the new aggregations.

Summary

This chapter introduced you to the Analysis Services component of SQL Server 2000. We started with a refresher on database history as we looked at the evolution of OLTP systems to also encompass data-warehousing capabilities in order to gain more value from their data. We discovered that OLAP is a means of accessing data from such data warehouses or their smaller counterparts, data marts. We also looked at the benefits of data mining when it comes to finding trends and patterns in large volumes of data and how data-mining techniques are being successfully used to develop prediction models. We capped off our database theory review with a look at the various structures of relational and dimensional databases and discovered that these differences are in place in order to allow each type of database to function more effectively in its role. For example, a relational database is great for speed and transactional throughput but would be effectively unusable in a very large data warehouse due to all the joins between tables required to produce summarized data.

This chapter also introduced some of the new features added as SQL Server 7's OLAP Services evolved into SQL Server 2000's Analysis Services. Data mining is the perhaps the biggest new entry; we also examined other less prominent but useful items such as new dimension types and general improved and broader functionality. Once we saw the new tools we had available to use, we covered the requirements and the process for installing Analysis Services. We then looked at components that make up Analysis Services, paying particular attention to Analysis Manager, which is a primary means of access to much of the functionality we used throughout the examples covered in this chapter.

We spent some time reviewing the cube, a multidimensional structure used to model subsets of data from data warehouses, because we recognized its key status within OLAP solutions. We looked at the elements of a dimensional database, the measures that represent the data that will be totaled and summarized, and the dimensions, such as time, over which the measures will be analyzed. We used the theory we learned to begin to build our own OLAP solution to gain familiarity with both the concepts and SQL Server's tools. Together we built a cube to allow us to analyze the 1998 sales data from the sample FoodMart database.

Having also learned the concepts behind data mining, we put SQL Server's new data-mining component to the test, using both the relational mining model and the OLAP mining model to demonstrate how we could mine data from either a relational database or from a cube within a dimensional database. We discussed the various storage methods available for use in our data-mining cubes and looked at the pros and cons of varying levels of data aggregation (or prior summarizing) of data within cubes, then examined the impact differing approaches would have on mining performance. We used the data-mining browser to look at the prediction results gained from a mining model that had been previously trained using known data. We saw how this tool gave us a quick and highly visible overview of data depth and importance of attributes with its use of color

coding but still allowed us to gain more detailed information about specific nodes, should we need it.

Having covered many of the capabilities of Analysis Services, we looked at a new feature allowing access over the Web utilizing HTTP. We saw that IIS is required to deliver this functionality and covered the steps required to set up the virtual directory and associated settings that provide access to SQL Server over HTTP. Finally, we concluded our discussion with a look a two performance monitoring and optimization tools—the Usage Analysis Wizard and the Usage Based Optimization Wizard—and saw how they could be used to identify and remove potential bottlenecks within a data warehouse.

FAQs

Q. I have to introduce the concept of data warehousing into my organization. I'm concerned about the complexity, cost, and amount of time it will take to do this. How can I go about this task without biting off more than I can chew?

A. If you have no data-warehousing facilities in place yet, consider introducing a data mart for a single department first. Because this operates on a smaller scale, it will be quicker, cheaper, and less difficult to get up and running. Later you can add data marts for other departments and gradually build up a complete data warehouse. However, be careful to still think of the big picture, because failure to consider how the data marts will fit and work together could lead to a problematic solution due to varying designs and structures between the data marts.

Q. I'm having trouble creating an OLAP cube with the Cube Wizard. In the step where I select my fact table, the Next button is grayed out and I can't proceed. What has gone wrong?

A. If you look at the bottom left of the Cube Wizard screen, you will see the words "The selected table lacks required numeric columns." All fact tables must have at least one numeric column, because they are designed to be summed across dimensions. Either add a numeric column to the table you would like to be the fact table or choose another table. It is possible to use OLAP to provide counts rather than sums. To do this, create a new (dummy) column in the fact table and set to 1 the value held by that column in every row. Then when OLAP sums across dimensions, it will also produce a count across dimensions.

Q. I need to set up a second Analysis Server, but when I try to register it, I get a message saying that I don't have enough permissions to administer the Analysis Server. I know Analysis Services supports multiple Analysis Servers, so how can I register my new server?

A. In order to register an Analysis Server, you must be logged on to that server with an account that is a member of the OLAP administrators group. Either log on to the server with an account that is a member of the OLAP administrators group or add your account to the group (on the new server) using User Manager. You need to log off and log back on again before you receive the desired access.

Chapter 11: Using XML with SQL Server

Introduction

Extensible Markup Language, or XML, has met with immense popularity with the continually increasing need for dispersed systems to communicate in a growingly connected environment. Developed in 1996 and recorded by the World Wide Web Consortium (W3C) in 1998, XML offers many advantages over complex and proprietary approaches to communication. True to its name of *extensible* , XML documents can be as simple as a single set of elements or as complex as necessary to accomplish nearly any task. Microsoft continues to make fundamental shifts in its technology architecture to support the processing, delivery, and use of XML, and many other organizations are quickly adopting it as an integral part of their solutions. With industry-standard support, extensible structure, and a low learning curve, XML will continue to increase its role and impact on technology solutions.

As one of the more publicized additions to SQL Server 2000, native XML support allows your organization to begin taking advantage of XML and prepare your applications for its increased use in the future. The addition of the FOR XML statement in T-SQL offers the straightforward delivery of XML-formatted result sets. Several new stored procedures are available for using XML documents as record sets in T-SQL. Client applications can take advantage of XML using the new extensions to ActiveX Data Objects (ADO) in version 2.6. SQL Server 2000's support of HTTP access allows remote applications to retrieve and manipulate SQL Server data as XML over the Internet, with very little additional configuration.

This chapter provides an overview of XML, its components, and its structure. You will configure support for HTTP server access using IIS and query SQL Server using a Web browser. Advanced XML topics such as XPath statements and XML views will introduce you to real-world XML solutions and walk you through creating the necessary components to utilize XML in client and server applications.

Overview of XML and SQL Server Support

In this chapter, we discuss XML and the support for XML that is included in Microsoft SQL Server 2000. We also discuss what XML is and some of the languages available to work with XML. This discussion includes XML documents, the Extensible Stylesheet Language (XSL), XML-Data Reduced (XDR) Schemas, and the XML Path Language (XPath). In addition, we discuss how XML data can be retrieved from SQL Server from a URL using HTTP and how to use the XML languages with SQL Server. All the examples in this chapter use following products: Microsoft Windows 2000 Server, IIS 5.0, SQL Server 2000, and Internet Explorer 5.5.

What Is XML?

The W3C describes XML as "the universal format for structured documents and data on the Web." XML was developed and standardized by the W3C. The W3C describes why it developed XML as follows: "XML is primarily intended to meet the requirements of large-scale Web content providers for industry-specific markup, vendor-neutral data exchange, media-independent publishing, one-on-one marketing, workflow management in collaborative authoring environments, and the processing of Web documents by intelligent clients" [971208 W3C press release, www.w3c.org/xml]. The W3C intended XML primarily for use in Web applications. However, the usefulness of XML has extended beyond that realm; it is also used in standard applications in Microsoft's .NET platform and for exchanging data among disparate systems. XML allows data to be exchanged regardless of the platform and the development language.

XML is an open standard that allows data to be shared universally on the Internet across any platform that supports XML. XML allows structured data to be described using a text-based format. These data can then be delivered and exchanged over the Hypertext Transport Protocol (HTTP), which is the protocol used on the Web. When XML data are received, the data can be used for display and/or manipulation on the client, or they can be used for data exchange between servers. XML is a simplified subset of the Standard Generalized Markup Language (SGML). Even though XML was designed for the Internet, it is not limited to Internet applications. It can be used in standard client/server applications as well. Microsoft's new .NET strategy leverages XML as its standard data exchange format.

The Benefits of XML

XML is designed to give Web applications more flexibility and power. A number of significant benefits are derived from using XML, such as better context searching, Web applications that are more flexible, open standards, and a format for delivering data on the Web.

Context searches on the Internet can be greatly enhanced using XML. With XML, data can be uniquely tagged to give them meaning. For example, say you are searching for books about Shakespeare. If you do a search on the Internet you will get over a million

hits. Many of these books will be about *books* written by Shakespeare. Wouldn't it be nice if the data were tagged so that you could simply return hits for books about Shakespeare himself? XML makes this possible. Of course, XML must be implemented in an unambiguous manner. Over time, standard tags will be created to facilitate more meaningful searches on the Internet.

XML allows for more flexible Web applications. It adds data typing and structure. It can allow for integration of data from data sources that are not compatible with each other. Data can be exchanged between and/or merged from disparate data sources without a predefined data format, because XML is self-describing. Once the XML data are received, you can parse the relevant data to transform and/or present the data in whatever manner you desire. This concept of separating the data from the presentation of the data allows for more powerful Web applications as well as multiple views of the same data without multiple trips to the server to retrieve the data. Updates to data can be accomplished without retransmitting the entire data set. Updates can be exchanged by exchanging only the data that were changed, greatly reducing the amount of data required to be transmitted over the wire.

XML is an open standard for the Internet to allow interoperability across platforms on the Web. This standard includes XML, XSL, XDR, and XPath. We will discuss these elements later in the chapter. XML data can be sent and received over the Internet using HTTP. This is the same protocol that delivers HTML, so existing network architectures can be used in their current configurations. You do not have to configure your proxy servers and firewalls to allow for XML data to be transmitted and received. XML data also compress well, given their text-based nature.

Working with XML

In this section, we take a look at how to use XML. We discuss XML documents and the rules for them. When exchanging data, the two sides involved in the exchange might want to create some standards for the data. This standard creation is done using *schemas* . This section covers schemas and the basics for creating them. XML data are raw data that are separate from the data's presentation. In HTML, you can use cascading style sheets (CSS) to format a Web page. In XML, you can use the XSL to format XML data for presentation and to transform XML data. Finally, we look at using XPath to retrieve subsets of data from an XML document.

XML Documents

Exactly what is an XML document? It is a data object that is well formed. Simply put, *well formed* means that the data object follows the rules of the document and the XML language. XML documents are composed of *entities* . Every document has a "root" entity that starts the document, but the document can also contain other entities that are included in the document. The document is composed of declarations, elements, comments, and processing instructions using a markup language.

Specifically, an XML document is the top level of abstraction for an XML data set. A document contains a single *named element* , commonly referred to as the *root element* . Elements can contain character data, attributes, sub-elements, and processing instructions. The *character data* are the values for the element. *Attributes* are uniquely named, with each attribute containing a string value. *Sub-elements* are child elements in a hierarchical tree. A *processing instruction* is an instruction to the receiver of the document on how to process the data. This could be any agreed-on instruction. A *comment* is just what it sounds like—a comment for the reader of the document. Comments are not processed or parsed.

XML documents can be well formed, valid, or not formed. A *well-formed* document conforms to the syntax that is defined by XML 1.0. A *valid* XML document is well formed and meets the specifications of its schema. A schema specifies the elements to be used, their order, and any attributes in the elements.

Some rules govern the composition of a well-formed XML document. These rules are similar to HTML rules except they are more stringent. Start and end tags must match and be strictly nested. Each start tag must have an end tag. You can have text and sub-elements within the start and end tags. This differs from HTML, in which, if you are missing an end tag, for example, the HTML parser will try to figure it out for you. In XML documents, you will get an error message if you are missing an end tag. Elements cannot overlap each other. This means that if an element has a sub-element start tag, the sub-element's end tag must come before the parent element's end tag. The following is an example of an invalid, overlapping element:

```
<FirstName>John
        <LastName>Doe
</FirstName>
        </LastName>
```

Another important rule to remember is that XML tags are case sensitive. For instance, *firstname* is different from *FirstName,* which is different from *Firstname* . It is valid to self-close a tag by including the slash within the start tag. The following is an example of a self-closing tag:

```
<FirstName/>
```

You also have to be careful about using reserved characters. These characters need to be substituted with nonreserved characters. Table 11.1 is a list of the reserved characters and their substitutions. Keep in mind that these are also case sensitive.

Finally, a well-formed XML document must have a root element. This element does not have to be named *ROOT* , but it commonly is.

Table 11.1: XML Reserved Characters

Reserved Character	**Substitution**

Table 11.1: XML Reserved Characters	
Reserved Character	**Substitution**
<	&lt
&	&amp
>	&qt
“	&quot
‘	&apos

Let’s take a look at what a well-formed XML document looks like. Here is an example:

```
<?XML version='1.0' encoding='UTF-8' ?>
<!—put a comment here, same syntax as HTML —>
<rowset xmlns='the xml namespace is specified here'>
  <row>
   <FirstName>Cameron</FirstName>



   <LastName>Wakefield</LastName>
 </row>
 <? Put a processing instruction here?>
 <row>
   <FirstName>Lorraine</FirstName>
   <LastName>Wakefield</LastName>
 </row>
</rowset>
```

Extensible Stylesheet Language

XSL is a language for creating style sheets for XML data. XSL is composed of XSLT for transforming XML documents and an XML vocabulary used to specify formatting of XML data. XSLT allows you to transform XML documents into other XML documents that can be formatted using the XML vocabulary. This system allows you to present the same XML document in different ways. XSL is similar to a style sheet such as CSS. (As a side note, an XML document can be formatted with CSS.)

XSL contains rules in a template form that matches patterns against nodes in the XML document. Let’s take a look at an example of an XSL file. This example transforms an XML document by displaying the first and last names of customers in an HTML table. The example uses the FirstName and LastName sub-elements of the Customers element. These elements are specified in the *<xsl:template match=’Customers’>* tag. Then in the HTML table is the tag *<xsl:apply-templates select=’root’/>,* which places the data at that location in the format specified by the template:

```
<?xml version='1.0' encoding='UTF-8'?>
    <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

```
     version="1.0">
    <xsl:template match = '*'>
        <xsl:apply-templates />
   </xsl:template>
   <xsl:template match = 'Customers'>
      <TR>
        <TD><xsl:value-of select = 'FirstName' /></TD>
        <TD><B><xsl:value-of select = 'LastName' /></B></TD>
      </TR>
   </xsl:template>
   <xsl:template match = '*'>
     <HTML>
      <BODY>
       <TABLE border='1' style='width:300;'>




        <TR><TH colspan='2'>Customers</TH></TR>
        <TR><TH >First name</TH><TH>Last name</TH></TR>
        <xsl:apply-templates select = 'root' />
       </TABLE>
      </BODY>
     </HTML>
  </xsl:template>
</xsl:stylesheet>
```

The following is an example of an XML document that could be used with the previous XSL style sheet. This example produces an HTML table that displays only the first and last names of the customers in an HTML table:

```
<?XML version='1.0' encoding='UTF-8' ?>
<Customers>
  <FirstName>Cameron</FirstName>
  <LastName>Wakefield</LastName>
  <City>Rockledge</City>
  <State>FL</State>
  </Customers>
<Customers>
  <FirstName>John</FirstName>
  <LastName>Doe</LastName>
  <City>Orlando</City>
  <State>FL</ State>
</Customers>
```

XML Data-Reduced Schemas

An XML schema is a model used to describe the structure of data in an XML document and to specify any constraints on the data. The schema is in human-readable format. It specifies the arrangement of tags and text to create a valid XML document. This allows XML parsers to verify that the XML document conforms to a standard (the schema). You can think of the schema as being like a table definition in a database. This structure

allows data to be verified to meet predetermined specifications before they are exchanged.

An XDR schema is opened and closed with a Schema tag. The tag can contain attributes to specify the schema name and the namespace for the schema. The following example specifies that the *FirstName* and *LastName* attributes are in the *Customers* element. At a minimum, you must declare the *xml-data* namespace.

```
<?xml version="1.0" ?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data">



<ElementType name="Customers" >
    <AttributeType name="FirstName" />
    <AttributeType name="LastName" />
    <attribute type="FirstName" />
    <attribute type="LastName" />
</ElementType>
</Schema>
```

XML Path Language

The XML Path Language (XPath) was created to define a syntax for addressing subsets of data within an XML document. XPath uses the logical structure of an XML document by using path notation similar to that of URLs. This path notation is used to navigate the hierarchical tree structure of an XML document. It allows you to select a set of nodes in an XML document.

There are four categories of XPath operators: Boolean (and, or), relational (<, >, <=, >=), equality (=, !=) and arithmetic (_, -, *, div, mod). XPath operators can be used in conjunction with each other so that an operator can select nodes based on the set of nodes selected by the previous operator. XPath also allows you to manipulate data and transform from one XML document type to another. XPath queries against annotated XDR schemas, which allows you to provide XML views of data. An XPath query can be executed either as part of a URL or within a template. The data that are returned are structured according to the XDR schema they are run against.

XPath uses the concept of an *XPath context* . You can think of this feature as resembling a cursor. It is a node that can be any node in the tree hierarchy. Basically, this means that your XPath queries will look for nodes relative to the context node in the tree. To query the entire document, you use the root node. Look at Figure 11.1 for an example of a tree structure and its XML equivalent.

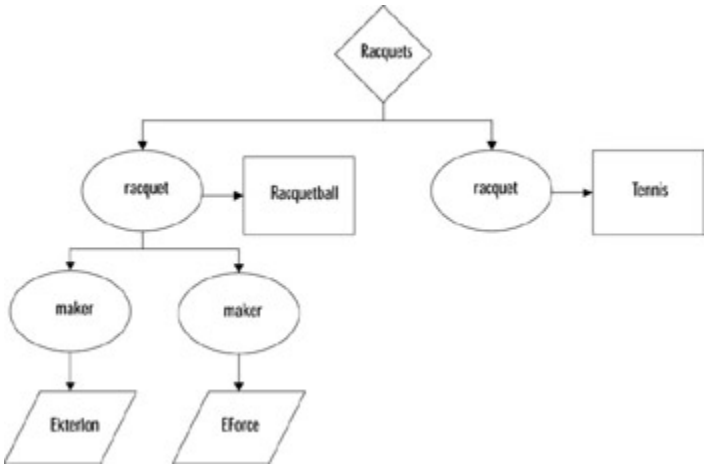


Figure 11.1: XML tree structure.

```
<?xml version="1.0"?>
<Racquets>
    <racquet type="Racquetball">
        <maker>Ektelon<maker/>
        <maker>EForce<maker/>
    <racquet/>
    <racquet type="Tennis">
    <racquet/>
<Racquets/>
```

The XPath expression for the racquet maker nodes would be:

```
/Racquets/racquet/maker
```

This is a *location path* . Location paths can be absolute or relative. When a location path begins with a forward slash (/), it is an *absolute path* starting with the root node. You can create *relative paths* by removing the leading forward slash. It will then be relative to the current *context node* . A context node can have various relationships to adjacent nodes (for example, parent to child). These relationships in XPath are referred to as *axes* . An axis is the direction in which you should look for data. Table 11.2 describes the XPath axes and their identifiers. Figure 11.2 shows how the axes are partitioned.

Table 11.2: XPath Axes and Identifiers

Axis Identifier	**Description**
Self	The context node itself.
Child	The context node’s direct children.
Parent	The context node’s direct parent; only one node unless root, then none.
ancestor or self	The context node’s parent and its parent, up to the root node.
Descendant or self	The context node’s children and their children down to the leaf nodes.

Table 11.2: XPath Axes and Identifiers

Axis Identifier	**Description**
Following [-sibling]	Nodes that come after the context node in the document, excluding descendant nodes.
Preceding	Nodes that come before the context node in the document, excluding ancestor nodes.
Attribute	The context node's attribute nodes.
Namespace	The context node's namespace nodes, including the default namespace.

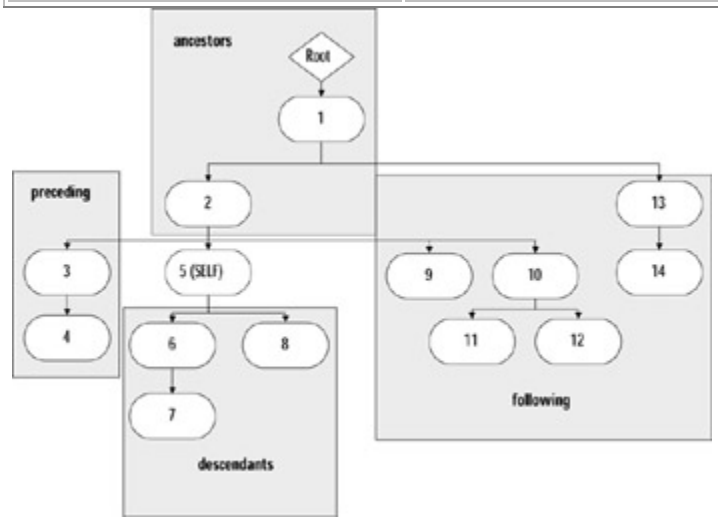


Figure 11.2: A partitioned XML tree structure.

You can use what is called a *node test* to filter nodes using axes by name or type. The syntax for a node test is shown in the following examples. The first example is a node test for the racquet nodes by name. The XPath expressions can also be combined to create longer XPath expressions. The expressions are separated by forward slashes. Each subsequent expression uses the preceding expression's result set for its context, ending in a single, final result set. The second example performs a node test on the racquet nodes but filters based on the *type* attribute of those nodes, using a predicate expression:

```
Evaluate by name:                child::racquet
Evaluate by attribute name:     child::racquet[attribute::type]
```

Predicates are used to further filter an XPath expression's result set. A predicate expression is appended to an XPath expression and enclosed in square brackets. The predicate expression is evaluated to TRUE or FALSE for each node in the XPath expression's result set. If it evaluates to true, the node remains in the result set. The following example retains only nodes that have a type attribute with a value of *Ektelon:*

```
Child::racquet[attribute::type='Ektelon']
```

XPath includes some functions that can be used for more complex expressions. Table 11.3 lists some of the more commonly used XPath functions.

Table 11.3: XPath Functions

XPath Function	Description
last()	Returns the index number of the last node in the context.
position()	Returns the position of the current context node in the current result set.
count(node-set)	Returns the number of nodes in the node set passed to it.
id(object)	Returns nodes in the XML document, selected by their unique ID.
string(object)	Converts the passed object to a string (for example, a Boolean value to TRUE).
starts-with(string, string)	Returns Boolean value of whether or not the first string starts with the second string.
translate(string, string, string)	Translates the first string by changing the characters in the second string to the characters in the third string. This is the only way to change case, for example.
number(object)	Converts the passed object to a number. A string is converted to a number. A Boolean value is converted to 0 or 1.
sum(node set)	Returns the sum of the numerical values of the nodes in the passed node set.
round(number)	Rounds a floating-point number to its nearest integer value.

XML Support and Limitations in SQL Server

Microsoft SQL Server 2000 has added built-in support for XML. It allows you to access SQL Server through a URL using HTTP. You can create XDR schemas and run XPath queries against them. You can retrieve data in XML format using the FOR XML clause in a SELECT statement. You can write XML data to SQL Server using the new OpenXML rowset provider. The SQL Server OLE DB provider has been enhanced to allow XML documents to be specified as command text and to return results in a stream. However, Microsoft SQL Server 2000 does not support all XML functionality. Not all XPath functionality is implemented, and XML data cannot be used in all SQL statements. Currently, a limited subset is implemented.

Additional XML Resources on the Web

The Internet has numerous other resources for XML. The official standards for XML are developed by the W3C. Biztalk, an industry initiative started by Microsoft, is a community of users of standards such as XML.

W3C.org

XML was developed by the W3C. The W3C was created in 1994 to develop common protocols and languages for the Web. It comprises over 400 member organizations. On the W3C Web site, you can find more information about the W3C and extensive information on XML, including new standards, the complete specifications, and the status of all the XML languages. You can also see the W3C press releases and just about any other information you can think of.

Biztalk.org

Biztalk is a community of standards users that is supported by a wide range of organizations. Biztalk has two core issues: first, that application integration is too complicated and costly; second, that the next phase of the Internet will require much greater application integration across disparate systems. Biztalk members are adopting an XML message-passing architecture that is platform neutral to tie systems together. Biztalk is committed to making XML interoperable by supporting standards and developing software tools to facilitate it.

XML.org

XML.org is an independent source of information and resources about using XML in industrial and commercial applications. It also provides some tools to help you determine how to use XML for your business. It is a vendor-independent organization to help with universal data exchange across all vendors. XML.org is hosted by the Organization for the Advancement of Structured Information Standards (OASIS). OASIS is an organization that creates interoperability industry specifications based on public standards such as XML.

MSDN.Microsoft.com/XML

You can obtain Microsoft-specific information about XML from the company's MSDN Web site, which shows highlights of new tools and features. It also contains the complete documentation for the XML SDK, sample code, downloads, and more. This site contains extensive information on using XML with Microsoft products.

HTTP and URL Query Support

Microsoft SQL Server 2000 has added functionality to allow data to be retrieved from a database through a URL using HTTP. This is the same protocol by which HTML is transmitted, which allows data to be transmitted without special configuration of the network architecture. To supply this feature, it works in conjunction with Microsoft Internet Information Services (IIS).

Configuring IIS for HTTP Query Support

Before you can access Microsoft SQL Server 2000 through HTTP, you must configure IIS. You must create an IIS virtual directory through which to execute SQL Server queries. If you want to implement templates and schemas, you must also create virtual names for them.

Creating a Database Virtual Directory in IIS

In order to run SQL Queries over HTTP using a URL, you are required to create a virtual directory in IIS and configure it to use an instance of SQL Server and a database on that instance.

Configuring IIS to Allow URL Queries

1. From the Start menu, click Programs | Microsoft SQL Server | Configure SQL XML Support in IIS.
2. Now we will create a new virtual directory called XMLDemo on IIS. Right-click Default Web Site. From the Context menu, select New | Virtual Directory. This action will bring up the New Virtual Directory Properties dialog.
3. In the Virtual Directory Name frame, enter **XMLDemo** .
4. In the Local Path field, enter **C:\INETPUB\XMLDemo** . You can use the Browse button to bring up the Browse dialog box or Windows Explorer to create this folder. Your settings should look like those in Figure 11.3 .



Figure 11.3: Virtual directory name and path settings.

5. On the Security tab, enter the username and password for your SQL Server (for example, username: **sa** with a blank password).
6. On the Data Source tab, select the SQL Server you will be using (for example, **local**) and select the Northwind database.
7. On the Settings tab, check all the settings.
8. On the Virtual Names tab, click the new button to bring up the Virtual Names Configuration dialog box.
9. For the Virtual Name, enter **Queries** , and select Template for the type. For the path, enter **C:\INETPUB\XMLDEMO** .
10. Click the Save button to save the new virtual name. The Virtual Names tab should now look like Figure 11.4 .

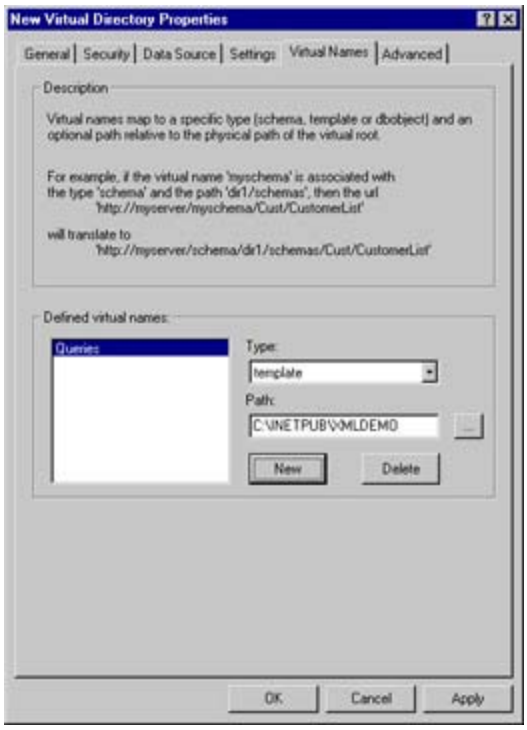


Figure 11.4: Adding a virtual name.

11. For the virtual name for schemas, on the Virtual Names tab, click the New button to bring up the Virtual Names Configuration dialog box.
12. Enter **Schemas** , and select Schema for the type. For the path, enter **C:\INETPUB\XMLDEMO** .
13. Click the Save button to save the new virtual name.
14. Click OK to create the new virtual directory. You will now have the XMLDemo virtual directory under the default Web site.

Querying SQL Server Using HTTP

You can execute queries against a SQL Server database using HTTP. Doing so allows you to view the data in the Web page by submitting a query in the URL. This feature is supported by the SQL ISAPI extensions. You must use the correct syntax when using HTTP to query SQL Server. The syntax for URLs are as follows:

```
http://iisserver/virtualroot/virtualname[/pathinfo][/XpathExpression]
[?param=value[&param=value]…n]
```

Or:

```
http://iisserver/virtualroot?{sql=SqlString | template=XMLTemplate}
[&param=value[&param=value]…n]
```

The arguments in the syntax are shown in Table 11.4 .

Table 11.4: URL Query Arguments

Keyword	Description
Iisserver	The IIS server domain name or IP address.
Virtualroot	The virtual root that is configured to access the database.
Virtualname	A virtual name configured in the virtual root.
Pathinfo	The path to the template file or mapping schema file.
XpathExpression	A virtual name of type dbobject or schema for the virtual root.
Sql	The SQL query. This can be a SQL query or stored procedure.

Supported Query Methods Using HTTP

With SQL Server 2000, you can query a database using HTTP. This task can be accomplished from the URL. In the URL, you can include the SQL query itself, specify a template file, or use XPath queries.

Executing SQL Using HTTP

There are several ways to query a database over HTTP. You can use a SQL query, templates, template files, or XPath queries, or you can specify a database object such as a table or view. All these methods can be included in the URL to retrieve a result set.

You can specify a SQL query in the URL statement to retrieve data. In order to do so, your IIS server must have the Allow URL Queries option set for the virtual directory. We did this in Exercise 11.1. When returning XML documents for which there is more than one element at the root level, a root element must be added. You can add a root element by prefacing the query with SELECT <ROOT> and ending it with SELECT </ROOT> or by passing the root keyword with a value in the query.

Let's take a look at how this process works. If we enter a query that returns only one row, we don't need the root element. If we perform the following query from Internet Explorer, it returns only one row, so the root element is not needed. This query is run against the XMLDemo virtual directory we created earlier in the chapter. Notice that all the spaces in the query are replaced with plus signs (+):

```
http://localhost/XMLDemo?sql=SELECT+TOP+1+CompanyName,+Phone+ _
                           FROM+Suppliers+FOR+XML
```

The result of that query is:

```
<row CompanyName="Alfreds Futterkiste" Phone="030-0074321" />
```

If you executed the preceding query but changed the query to return the first two rows, you would receive an error message, as shown in Figure 11.5 .

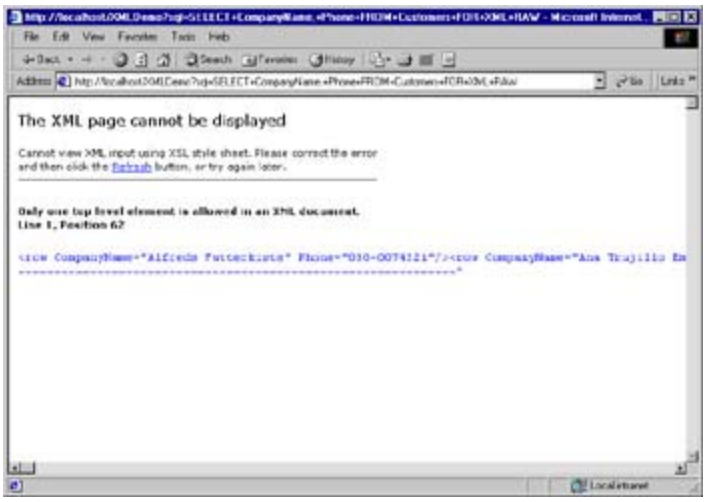


Figure 11.5: The result set with a ROOT element error.

So, let's see how to add the root element. First, we add it using the SELECT <ROOT> statement. The URL is:

```
http://localhost/XMLDemo?sql=SELECT+'<ROOT>'; _
SELECT+TOP+2+CompanyName,+Phone+FROM+Customers+FOR+XML+RAW; _
SELECT+'</ROOT>'
```

You can also add the root element by passing the keyword *root* . This URL is:

```
http://localhost/XMLDemo?sql=SELECT+TOP+2+CompanyName,+Phone+ _
FROM+Customers+FOR+XML+AUTO&root=ROOT
```

The result set follows. Notice that the root element is now part of the XML document:

```
<ROOT>
  <row CompanyName="Alfreds Futterkiste" Phone="030-0074321" />
  <row CompanyName="Ana Trujillo Emparedados y helados"
       Phone="(5) 555-4729" />
</ROOT>
```

You can also query the database without returning XML. However, you are severely limited in this activity. This is considered streaming data, so you are allowed to return only one column. You can do this by removing the root element and the FOR XML clause, as in the following example:

```
http://localhost/XMLDemo?sql=SELECT+CompanyName+FROM+Customers
```

Creating an XML Query Template

SQL queries can be very long and complex, which makes them difficult to enter in a URL and therefore prone to error. A URL is also limited to 1KB (1024 characters) in length. To help alleviate this problem, you can use *template files* . Template files can

contain queries for SQL or XPath. To execute a template file, you must specify the template file in the URL.

Templates are well-formed XML documents that contain a SQL statement or statements or XPath query or queries. Templates are somewhat flexible and allow you to perform a variety of functions. You can execute SQL and XPath queries in template files. When an XPath query is executed, an XDR schema file must be specified in the template. You can define parameters to be passed to the SQL or XPath query. Namespaces can also be declared in the template. XSL style sheets can be applied to the XML document that is created. By specifying the top-level or root element for the XML document being created, you can make it a well-formed or valid document.

You can think of a template file as being similar to a stored procedure. You get some of the same benefits from a template file that you get with a stored procedure. You can verify the queries and make them available for everyone to use, or you can create parameters to pass to make them more flexible. Another benefit is that you get better security using templates, thus preventing the queries from being edited in the URL and from allowing users to see what the query is. You can also turn off the option allowing URL queries and allow only the XML ISAPI to process files.

You can use XML templates to specify how SQL queries and XPath queries are used. Templates are valid XML documents that contain SQL statement(s). These templates allow you to define how data are retrieved in a valid XML document. The following is the syntax for an XML template. Note that all the elements in the template syntax except for the namespace declaration are optional:

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql"
       sql:xsl='XSL FileName' >
  <sql:header>
    <sql:param>..</sql:param>
    <sql:param>..</sql:param>…n
  </sql:header>
  <sql:query>
    sql statement(s)




   </sql:query>
   <sql:xpath-query mapping-schema="SchemaFileName.xml">
     XPath query
   </sql:xpath-query>
</ROOT>
```

For a complete listing of XML template tags, refer to Table 11.5 .

Table 11.5: XML Template Tags

Tag	Description

Table 11.5: XML Template Tags

Tag	Description
ROOT	Provides the root tag. This tag does not have to be ROOT, it can have any value.
sql:xsl	Allows you to specify an XSL style sheet to be applied to the resulting XML document.
sql:header	Contains header information. Currently, this tag is limited to the <sql:param> element. You can define multiple parameters as sub-elements to this tag.
sql:param	Allows you to define a parameter to be used in the query in the template. Each element can define only one parameter.
sql:query	Specifies SQL queries. You can have more than one of these elements in a template.
sql:xpath-query	Specifies an XPath query. This tag will be executed against an XDR schema, so the schema filename must be included.
mapping-schema	Specifies an annotated XDR schema for an XPath query.

When you create a template, it must be stored in the relative path of the physical directory for a virtual name of type *template* . This means that it can be in that directory or one of its subdirectories. Let's take a look at the following example of an XML template file. This example performs the same query we used previously to retrieve the company name and phone from the Suppliers table:

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
    <sql:query>
      SELECT TOP 2 CompanyName, Phone
      FROM Suppliers
      FOR XML AUTO
    </sql:query>
</ROOT>
```

Executing XML Query Templates

To execute an XML template, you must specify the template filename in the URL. Remember that the virtual name must be of type *template* for this to work. Let's look at an example that calls an XML query template file. If we create a template file with the query template shown in the previous section and name it suppliers.xml, the following is how you will view its results from the URL. Notice that we use the virtual directory XMLDemo with the virtual name of type template *queries* . Below the URL is what is returned from the template file:

```
http://localhost/XMLDemo/queries/suppliers.xml
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <Suppliers CompanyName="Exotic Liquids" Phone="(171) 555-2222" />
  <Suppliers CompanyName="New Orleans Cajun Delights" Phone="(100)
```

```
555-4822" />
</ROOT>
```

Specifying Query Parameters

Now let's look at passing parameters to queries. When we created our template file in the previous section , we simply returned the top two rows. What if we wanted to search for particular companies by name? We can accomplish this task using *parameterized queries* . You can use the <*sql:header*> element to specify parameters and give it a default value. In the following template query, we will add a parameter named CompanyName with a default value of *%* to return all companies if no parameter is specified:

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <sql:header>
    <sql:param name="CompanyName">%</sql:param>
  </sql:header>


    <sql:query>
      SELECT CompanyName, Phone
      FROM Suppliers
      WHERE CompanyName LIKE @CompanyName + '%'
      FOR XML AUTO
    </sql:query>
</ROOT>
```

Now let's execute the template file. We will search for company names such as *exotic* . The following is the URL to execute the template file with a parameter. Notice that the parameter is the same as passing parameters in HTML URLs. After the filename, we add a question mark, followed by the parameter name and its value. Under the URL is the result of the query:

```
http://localhost/XMLDemo/queries/suppliers.xml?CompanyName=exotic
```

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <Suppliers CompanyName="Exotic Liquids" Phone="(171) 555-2222" />
</ROOT>
```

XPath Queries

You can execute XPath queries against XDR schemas similarly to the way you execute SQL queries against views. This process allows you to create an XML view, which is similar to a view in SQL Server. In this way, you can predefine the data set that is available for querying. In this section, we look at XPath queries and some of their limitations in SQL Server.

Overview of XPath Queries and SQL Server Limitations

XPath queries can be executed in several ways. You can specify XPath queries in a template against an XDR schema. You can use inline mapping schemas by including an annotated XDR schema directly in a template. Alternatively, you can map a schema in the URL for the XPath query to execute against. Microsoft SQL Server 2000 does not support all the W3C XML XPath specification. In the next section , we discuss the XPath limitations in SQL Server.

It is important to know that the relational operators in XPath specifications convert their operands to numbers. This means that you cannot do string comparisons. XPath doesn't inherently perform date comparisons, so SQL Server 2000 varied the XPath specifications to accommodate it. When relational operators compare strings, it performs a string operation rather than a number comparison.

Let's take a look at the XPath functionality that is implemented in SQL Server 2000. Table 11.6 lists the functions that are implemented and their descriptions.

Table 11.6: XPath Functions Implemented in SQL Server 2000

Feature	**Item**
Axes	attribute, child, parent, self
Boolean-valued predicates	Including successive and nested predicates
All relational operators	=, !=, <, <=, >, >=
Arithmetic operators	+, -, *, div
Explicit conversion functions	number(), string(), Boolean()
Boolean operators	AND, OR
Boolean functions	true(), false(), not()
XPath variables	

XPath Limitations in SQL Server

As mentioned previously, not all the XPath specification functionality is implemented in Microsoft SQL Server 2000. Table 11.7 shows a listing of the XPath functions that are not implemented.

Table 11.7: XPath Functions Not Implemented in SQL Server 2000

Feature	Item
Axes	ancestor, ancestor-or-self, descendant, descendant-or-self (//), following, following-sibling, namespace, preceding, preceding-sibling
Numeric-valued predicates	All
Arithmetic operators	mod
Node functions	ancestor, ancestor-or-self, descendant, descendant-or-self (//), following, following-sibling, namespace, preceding, preceding-sibling
String functions	string(), concat(), starts-with(), contains(), substring-before(), substring-after(), substring(), string-length(), normalize(), translate()
Boolean functions	lang()
Numeric functions	sum(), floor(), ceiling(), round()
Union operator	all

Additional Information on the XML Path Language Specification

If you want more information on using XML and XPath with Microsoft SQL Server 2000, you can go to the Microsoft MSDN Web site at http://msdn.microsoft.com/sqlserver . You can also view the complete XPath specification at the W3C.org Web site.

XPath Data Types and Conversions

XPath uses different data types from both SQL Server and XDR. XPath uses only the *string* , *number* , and *Boolean* data types. This means that XPath must also perform some data conversions. The XPath operators convert the operands differently. Implicit and explicit data type conversions can produce subtle changes to the result. Arithmetic operators convert their operands to numbers. Boolean operators convert their operands to Boolean. Relational and equality operators convert their operands to create a Boolean result. Table 11.8 shows how XPath, XDR, and SQL Server data types are converted.

Table 11.8: XPath to XDR to SQL Server Data Type Conversions

XDR Data Type	Equivalent XPath Data Type
bin.base64, bin.hex	N/A
Boolean	boolean
number, int, float, i1, i2, i4, i8, r4, r8ui1, ui2, ui4, ui8	number
id, idref, idrefs, entity, entities, enumeration notationnmtoken, nmtokens, char, string dateTime,	String

Table 11.8: XPath to XDR to SQL Server Data Type Conversions	
XDR Data Type	**Equivalent XPath Data Type**
dateTime.tz, uri, uuid	
fixed14.4	N/A
Date	String
Time, time.tz	String

Using XPath Queries

Now let's take a look at XPath queries and how to execute them. An XPath query can either be specified in a URL or it can be put in a template file that is run from a URL. You can even pass parameters to XPath queries. To execute an XPath query against an annotated XDR schema, the schema file is named in the URL or in the template. Remember that to execute a query against a schema, you must first create a virtual name of type *schema* . The XDR schema file must located in the directory associated with the virtual name or one of its subdirectories. Let's look at the syntax for specifying XPath queries in the following URL:

```
http://IISServer/VirtualRoot/SchemaVirtualName/SchemaFile/
XPathQuery[?root=ROOT]
```

Notice the root parameter at the end. This parameter provides us with our root element that is required for XML documents with more than one element. SchemaVirtualName is a virtual name in the virtual root of type *schema* . The SchemaFile is just what it sounds like—the XDR schema filename. XPathQuery is the XPath query to execute. If we follow the examples we used in the previous sections, the URL will look like the first example that follows. But what if the file were in a subdirectory? The second example shows the URL if the XDR schema file were located in the ABC subdirectory:

```
http://localhost/XMLDemo/Schemas/SuppliersSchema.xml/

XpathQuery


http://localhost/XMLDemo/Schemas/ABC/SuppliersSchema.xml/

XPathQuery
```

Let's see how to use an actual XDR schema. The following is a schema file that is mapped to the Northwind Suppliers table. This schema file will be named SuppliersSchema.xml and located in the directory associated with the schema's virtual name:

```
<?xml version="1.0" ?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
         xmlns:dt="urn:schemas-microsoft-com:datatypes"
```

```
        xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <ElementType name="Suppliers" >
    <AttributeType name="CompanyName" />
    <AttributeType name="Phone" />

   <attribute type="CompanyName" />
   <attribute type ="Phone" />
  </ElementType>
</Schema>
```

You can execute XPath queries from a template file. The following is an example of an XPath template file. It retrieves the XML nodes Suppliers. This XPath template file is named SuppliersXPath.xml:

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
    <sql:xpath-query mapping-schema="SuppliersSchema.xml">
        /Suppliers
    </sql:xpath-query>
</ROOT>
```

Now let's look at executing an XPath query against this XDR schema file. We will query for the Suppliers nodes. The first URL executes an XPath query from the URL, and the second URL executes an XPath query from a template file. The partial results are listed below the URLs.

```
http://localhost/XMLDemo/Schemas/SuppliersSchema.xml/
                                        Suppliers?root=ROOT

http://localhost/XMLDemo/Queries/SuppliersXPath.xml
<ROOT >
  <Suppliers CompanyName="Exotic Liquids" Phone="(171) 555-2222" />
  <Suppliers CompanyName="New Orleans Cajun Delights" Phone="(100)
                                                          555-4822"
/>
</ROOT>
```

SELECT…FOR XML

Microsoft SQL Server allows you to retrieve data from a database in XML format rather than as rowsets. You can use the new FOR XML clause of the SELECT statement. You can also specify the format of the XML document that is returned.

FOR XML Syntax and Use

When executing queries against a SQL Server database using HTTP, you can return XML documents using the FOR XML syntax. The syntax for using the FOR XML clause follows:

```
FOR XML mode [, XMLDATA] [, ELEMENTS][, BINARY BASE64]
```

Refer to Table 11.9 for a description of each syntax argument.

Table 11.9: FOR XML Syntax Arguments

Keyword	Description
Mode	Specifies the format of the XML document being returned. This can be RAW, AUTO, or EXPLICIT.
XMLDATA	Specifies that an XML schema is to be returned. The schema is included at the beginning of the XML document that is returned.
ELEMENTS	Specifies that the columns are returned as sub-elements rather than as attributes to the row element. This is used only in AUTO mode.
BINARY	Specifies that the data be returned in base64-encoded format to BASE 64retrieve binary data.

You can use the mode argument to specify the hierarchy of the data in the XML document. You can choose from three modes: RAW, AUTO, and EXPLICIT. In RAW mode, the result of the query is returned by transforming each row in the result set into an XML element and providing an identifier row as the element tag.

Let's take a look at a result set using the RAW mode. If we were to query the Northwind database Suppliers table for the supplier's company name and phone numbers in the RAW mode:

```
SELECT companyname, phone FROM suppliers FOR XML RAW
```

the following results would be returned. Notice that the element tag is *row,* and within the element each column in the query is an attribute of the element, and the attribute value is the column's value. The element is self-closing, so you do not need the closing element tag:

```
<row CompanyName="Exotic Liquids" Phone="(171) 555-2222" />
```

```
<row CompanyName="Mayumi's" Phone="(06) 431-7877" />
```

If we perform a query against multiple tables in RAW mode, the result is still returned as one element. For example, if we query the Products table for the products and join it to the name of the supplier:

```
SELECT productname, companyname FROM products LEFT JOIN suppliers ON
products.supplierid = suppliers.supplierid FOR XML RAW
```

a row is returned, as follows:

```
<row ProductName="Chai" CompanyName="Exotic Liquids" />
<row ProductName="Konbu" CompanyName="Mayumi's" />
```

Notice that all the columns are in the same element. Let's see how the other modes differ.

In AUTO mode, the result of the query is returned by creating XML elements for each table being queried. When querying a single table, nesting is not necessary. If we executed the same query as previously:

```
SELECT companyname, phone FROM suppliers FOR XML AUTO
```

but specify Auto mode, a row of the result is shown as follows. It looks very similar to our earlier result. However, instead of the element identifier being row, it is the name of the table, Suppliers:

```
<Suppliers CompanyName="Exotic Liquids" Phone="(171) 555-2222" />
<Suppliers CompanyName="Mayumi's" Phone="(06) 431-7877" />
```

If we perform a query against multiple tables in AUTO mode, the result is returned in a nested XML tree. For example, if we query the Products table for the products and join it to the name of the supplier:

```
SELECT productname, companyname FROM products LEFT JOIN suppliers ON
products.supplierid = suppliers.supplierid FOR XML AUTO
```

a row would be returned, as shown here:

```
<products productname="Chai">
  <suppliers companyname="Exotic Liquids"/>
</products>
<products productname="Chang">
  <suppliers companyname="Exotic Liquids"/>
</products>
```

Notice that all the columns are in the same element as the table to which they belong and that the result is in a tree format. The hierarchy of the tree is based on the order of the tables to which the columns in the SELECT clause belong. This format allows you to set the tree hierarchy by the way you order the columns. The first column is the top element

in the XML document returned. The second column is a sub-element of the top element, and so forth. If a later column belongs to a table already referenced, it is added as an attribute to the element for the table.

When using the AUTO mode, you can also have the result set return the columns as separate elements for each column rather than as attributes to the same element. This is done by adding the option ELEMENTS:

```
SELECT companyname, phone FROM suppliers FOR XML AUTO, ELEMENT
```

By adding the ELEMENTS option, the result is changed, as shown here:

```
<Suppliers>
  <CompanyName>Exotic Liquids</CompanyName>
  <Phone>(171) 555-2222</Phone>
  </Suppliers>
<Suppliers>
  <CompanyName>New Orleans Cajun Delights</CompanyName>
  <Phone>(100) 555-4822</Phone>
</Suppliers>
```

In EXPLICIT mode, you can completely control how the XML document is returned. However, you must provide more information to specify the XML document structure to be returned. The SELECT statement must be written to produce a correctly formatted rowset. It must create two metadata columns. The first column must be a named tag number, which stores the number of the current element. It is of type integer. The second column must be a named parent tag number for the parent element. It is also of type integer, but it can be NULL. This creates the tree hierarchy.

When specifying the columns, you must also specify the name of the row and the name of the sub-element. Let's look at an example using the query to retrieve the company name and phone number from the Suppliers table. The first column, 1 , is for the top level in the hierarchy; the second column is the tag for the parent. This is NULL since the top level doesn't have a parent. These are the two required columns. The third column is for the company name. The AS clause specifies the name of the row (Supplier), the level in the hierarchy (1), and the name of the element (CompanyName). The following example can be saved as a query template file and executed to return the results set in the specified format:

```
<root xmlns:sql="urn:schemas-microsoft-com:xml-sql" >
  <sql:query>
       SELECT 1 as Tag,
              NULL as Parent,
              CompanyName as [Supplier!1!CompanyName],
              Phone as [Supplier!1!Phone]
       FROM Suppliers
       FOR XML EXPLICIT
   </sql:query>
</root>
```

To execute this query, you use the same format as for the other modes. The following example executes the same query using HTTP and returns the results of the query:

```
http://localhost/XMLDemo?sql=SELECT+'<ROOT>'; _
SELECT+1+as+Tag,NULL+as+Parent, _
CompanyName+as+[Supplier!1!CompanyName], _
Phone+as+[Supplier!1!Phone]+ _
FROM+Suppliers+FOR+XML+EXPLICIT; _
SELECT+'</ROOT>'
```

```
<ROOT>
  <Supplier CompanyName="Exotic Liquids" Phone="(171) 555-2222" />
  <Supplier CompanyName="Mayumi's" Phone="(06) 431-7877" />
</ROOT>
```

Limitations of FOR XML

When using the FOR XML clause, there are some limitations that you must consider. For one thing, FOR XML is valid only in a SELECT statement. You cannot use it in any subselect statements, such as SELECT statements in a WHERE clause. This includes UPDATE, INSERT, and DELETE statements. You cannot use FOR XML in COMPUTE BY or FOR BROWSE clauses. It is not supported in AUTO mode in GROUP BY or aggregate functions. It is not valid in CREATE VIEW statements. FOR XML cannot be used with cursors. It cannot be used in an INSERT statement within a stored procedure.

Names used in SQL Server that are invalid, such as spaces, in XML map to escaped numeric encoding. XML names must begin with an alphabetic character, a colon (:), or an underscore (_). Since colons are used to start namespaces, it’s a good idea to use the underscore as the escape character.

XML Views

XML Views specify a mapping of a relational database from an XML document. They also limit the data that can be queried to only the data specified. This mapping is performed using the XDR schema annotations, which allow you to use XPath queries to retrieve data from the database in an XML document. This is similar to creating a view in a database.

XML Data-Reduced Schemas

XDR schemas are used to define XML views and specify the structure of an XML document, as well as map XML data to database tables, views, and columns. Microsoft SQL Server 2000 has added a number of annotations to the XDR schema language. They allow you to specify the mapping between the XML data and the database. By default, an element name maps to a table or view, and an attribute maps to a column. Annotations allow you to change the default mappings as well as specify the hierarchy of the XML document. Table 11.10 lists the annotations and their descriptions.

Table 11.10: XDR Schema Language Annotations

Annotation	**Description**
sql:relation	Maps XML data to a table.
sql:field	Maps XML data to a column.
sql:is-constant	Creates an XML element that does not map to any table. It creates an element with a constant value.
sql:map-field	Excludes schema items from the result.
<sql:relationship>	Specifies relationships between XML elements to create the tree hierarchy.
sql:limit-field	Limits the values returned based on a limiting value. sql:limit-value
sql:key-fields	Creates a column(s) to uniquely identify the rows in a table.
sql:target-namespace	Moves the elements and attributes from the default namespace to a different namespace.
sql:id-prefix	Creates valid XML ID, IDREF, and IDREFS. It prefixes the values of ID, IDREF, and IDREFS with a string.
sql:use-cdata	Specifies CDATA sections.
sql:url-encode	Returns a URI that can be used later for BLOB data.
sql:overflow-field	Identifies the database column that contains the overflow data.

In addition to annotating a schema for SQL Server, you can specify data types for elements and attributes in an XDR schema. The standard data types use the following namespace. It can then be added to the schema namespace:

```
urn:schemas-microsoft-com:datatypes


<Schema xmlns="urn:schemas-microsoft-com:xml-data"
        xmlns:dt="urn:schemas-microsoft-com:datatypes">
```

By specifying a data type for an element or attribute, you cause the data to be output in XML in that data type when retrieved from the database. You can accomplish this with either the dt or sql annotations when mapping between XDR and Microsoft SQL Server 2000. The dt:type syntax is used to specify an XML data type for attributes and elements. To map XML data to specific SQL Server data, the sql:datatype syntax is used. This specifies the data type of a column to which an attribute maps. When Microsoft SQL Server 2000 returns an XML document, the data types are converted to strings. Some data types require additional conversions. These conversions are listed in Table 11.11 .

Table 11.11: Data Type Conversions

XML Data Type	**SQL Server Conversion**
Bit	CONVERT(bit, COLUMN)
date	LEFT(CONVERT(nvarchar(4000), COLUMN, 126), 10)
fixed.14.4	CONVERT(money, COLUMN)
id/idref/idrefs	id-prefix + CONVERT(nvarchar(4000), COLUMN, 126)
time/time.z	SUBSTRING(CONVERT(nvarchar(4000), COLUMN, 126), 1+CHARINDEX(N'T', CONVERT(nvarchar(4000), COLUMN, 126)), 24)
Nmtoken/nmtokens	id-prefix + CONVERT(nvarchar(4000), COLUMN, 126)

Microsoft SQL Server 2000 also has a default mapping from SQL Server data types to XML data types. These mappings are shown in Table 11.12 .

Table 11.12: Mapping Between SQL Server and XML Data Types

SQL Server Data Type	**XML Data Type**
Bigint	i8
Binary	bin.base64
Bit	boolean
Char	char
datetime	datetime
decimal	r8
Float	r8
Image	bin.base64
Int	int

Table 11.12: Mapping Between SQL Server and XML Data Types

SQL Server Data Type	XML Data Type
Money	r8
Nchar	string
Ntext	string
nvarchar	string
numeric	r8
Real	r4
smalldatetime	datetime
smallint	i2
smallmoney	fixed.14.4
sysname	string
Text	string
timestamp	ui8
Tinyint	ui1
varbinary	bin.base64
Varchar	string
uniqueidentifier	Uuid

Let's look at an example of how to annotate XML data types. We will look at two date attributes: *OrderDate* and *ShippedDate* . The OrderDate attribute will not be annotated, therefore SQL Server will return the *datetime* data type. For the ShippedDate attribute, we will specify the *date* XML data type, so only the date portion will be included in the XML document.

Take a look at the following example. Notice that the element is mapped to the Orders table using the sql:relation attribute. In order to specify data types, an attribute must be mapped to a SQL Server column. The dt namespace is used in the following example for the XML data types:

```
<?xml version="1.0" ?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
         xmlns:dt="urn:schemas-microsoft-com:datatypes">

   <ElementType name="Order" sql:relation="Orders">
     <AttributeType name="OrderDate" />
     <AttributeType name="ShippedDate" dt:type="date" />

     <attribute type ="OrderDate" />
     <attribute type ="ShippedDate" />
   </ElementType>
</Schema>
```

Now let's look at an example of how to annotate SQL Server data types. This time we will look at mapping the XML *string* data type. An XML *string* data type can map to a number of SQL Server data types, including *nchar, ntext, nvarchar, varchar,* and *text* . We will use the sql:datatype syntax to explicitly map it to a *varchar* SQL Server data type. For this example, we will use the FirstName and LastName attributes. The FirstName attribute will not be mapped, and the LastName attribute will be. In the following example, notice that the elements are mapped to columns using the sql:field attribute. In order to specify data types, an attribute must be mapped to a SQL Server column. The sql namespace is used for the SQL Server data types:

```
<?xml version="1.0" ?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
    xmlns:sql="urn:schemas-microsoft-com:xml-sql">

   <ElementType name="Order" sql:relation="Orders">
     <attribute type="fname" sql:field="FirstName" />
     <attribute type="lname" sql:field="LastName"
                              sql:datatype="varchar" />
       </ElementType>
</Schema>
```

You can extend schemas to define nodes more specifically. You can type elements in a schema and extend data types by defining your own. You can also add comments to your schemas to explain complicated structures or data types. In addition to comments, you can use the description element anywhere in the schema that elements are allowed. It merely takes a start and end tag named *description* . See the following example for how to use extended data types and descriptions:

```
<ElementType name="SaleAmt" ext-dt:type="money"
              xmlns:ext-dt="urn:extensions:datatypes">
  <description>The money data type can have no more than 2
                decimal places.
  </description>
</ElementType>
```

Mapping XML Data to Database Tables and Columns

XDR schemas are used to map XML data to the database tables and columns. By default, an element maps to a table or view with the same name as that element. Attributes map to columns with the same name. There could be times when you need to change the defaults. You might need to change names, or you might need to map elements to columns rather than a table. If you want to map an element to a column, use the content attribute for the element with a value of textonly. You can also use the sql:field annotation, as discussed earlier in this chapter.

In the following example, we will look at the default mapping. The element name is Suppliers, which maps to a table name in the Northwind database. The attributes map to columns in the Suppliers database:

```
<?xml version="1.0" ?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
         xmlns:dt="urn:schemas-microsoft-com:datatypes"
         xmlns:sql="urn:schemas-microsoft-com:xml-sql">

 <ElementType name="Suppliers" >
   <AttributeType name="CompanyName" />
   <AttributeType name="Phone" />

   <Attribute Type ="CompanyName" />
   <Attribute Type ="Phone" />
 </ElementType>
</Schema>
```

SQL Server XML View Mapper

Microsoft has released a tool that will assist you in creating an XML View schema file with which to map an XDR schema to a SQL Server schema. This tool, still in beta testing at the time of this writing, is designed to work with the new XML functionality built into SQL Server 2000. It helps to speed the process of writing an XML View schema by hand and automates much of the process of merging the two schemas.

Downloading the SQL Server XML View Mapper

To use this tool, you must download it from the Microsoft Web site at:

http://msdn.microsoft.com/downloads/default.asp?URL=/code/sample.asp?url=/msdn-files/027/001/443/msdncompositedoc.xml

The following steps outline an example of how to use the SQL Server XML View Mapper tool.

Using the SQL Server View Mapper

1. First, start the tool. From the Start button, select Programs | Microsoft SQL Server XML Tools | XML View Mapper.
2. You will see a dialog box asking you to create a new project or open an existing project. Choose “Create a new XML View Mapper project,” and click the OK button. This action brings up a blank project, as shown in Figure 11.6 .

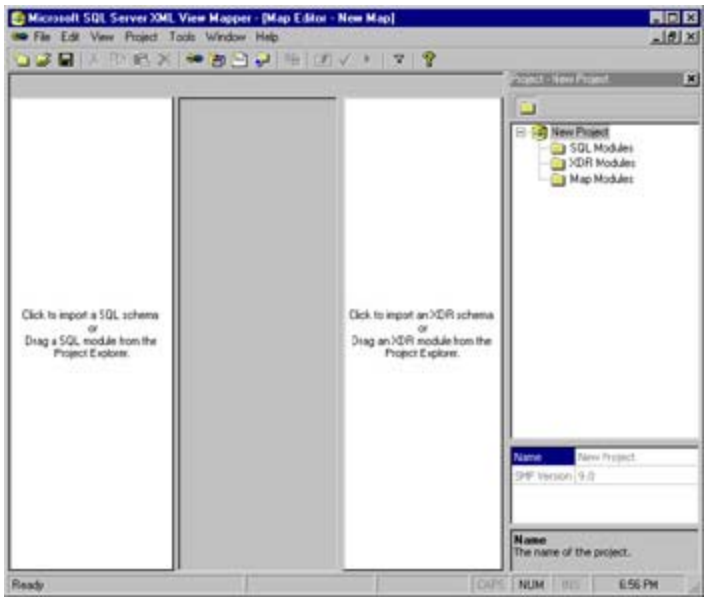


Figure 11.6: A new project in the SQL Server View Mapper.

3. Now let's add a SQL Schema. Click the left pane to import a SQL Schema.
4. Enter **localhost** or the name of your SQL Server for number 1. For number 2, enter your username and password (by default, the username is **sa** with no password). Then select the Northwind database in number 3. Your dialog box should look similar to the one in Figure 11.7 .



Figure 11.7: Data Link Properties.

5. Click the OK button to accept the settings.
6. Next you will choose the table(s) for your schema. Double-click the Suppliers table to select it, and click the OK button.
7. Now click the right pane to import an XDR schema. This will bring up a File Open dialog box.

8. Earlier in the chapter, we created an XDR schema and named it SuppliersSchema.xml. Browse to this file location, and click the Open button. If you did not create the file earlier, create it now with the following text and save it as SuppliersSchema.xml.

```
<?xml version="1.0" ?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
        xmlns:dt="urn:schemas-microsoft-com:datatypes"
        xmlns:sql="urn:schemas-microsoft-com:xml-sql">

  <ElementType name="Suppliers" >
    <AttributeType name="CompanyName" />
    <AttributeType name="Phone" />

    <attribute type="CompanyName" />
    <attribute type ="Phone" />
  </ElementType>
</Schema>
```

21. Notice that a mapping between the SQL schema and the XML schema is created automatically. Your project should look similar to Figure 11.8 .

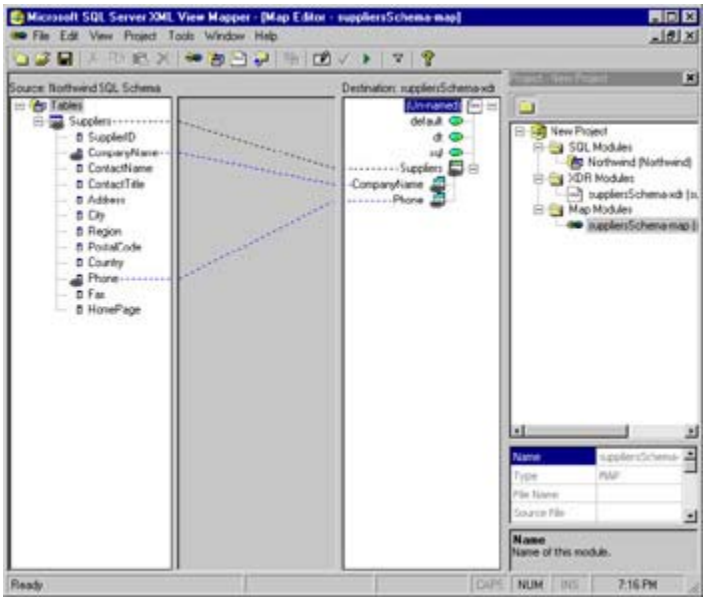


Figure 11.8: A SQL Server XML View Mapper project.

22. From this mapping, we can now create an XDR schema file. From the Tools menu, select Export XDR Schema. This will bring up a standard Save dialog box.
23. Enter the filename **suppliersSchema-map** , and click the Save button. That's it—the XDR schema file has been created. Note that you might want to open the file in Notepad to view the mapping.
24. The SQL Server XML View Mapper also allows you to view the results of XPath queries against the XDR schema. From the Tools menu, select XPath Query… or press F5.
25. First you see the Schema Load Log. Press the OK button.

26. Now you will see the XPath Query dialog box. In the XPath Query input box, enter /Suppliers, and press the Execute button. Your dialog box should look similar to the one in Figure 11.9 .

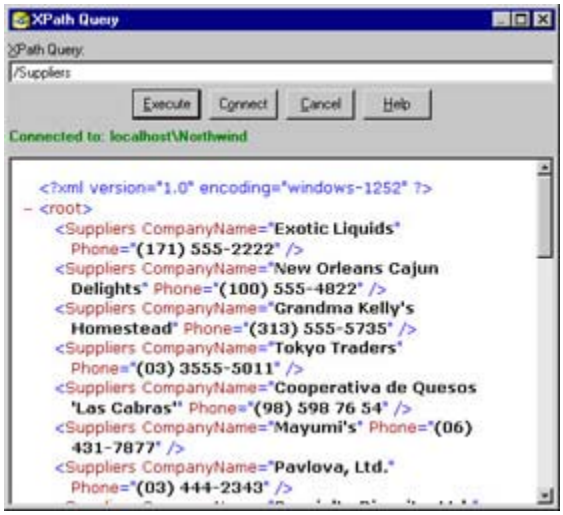


Figure 11.9: Using the XPath Query tool.

27. To close the program, click the Cancel button, then from the File menu, select Exit.

Using and Updating XML Data

Microsoft SQL Server 2000 has added several features to aid in working with XML. Instead of returning rowsets, SQL Server can return XML documents. You can execute these queries directly using the FOR XML clause or using stored procedures. This allows you to use XML in Web applications and client applications. Since XML was designed for use on the Internet and browsers have built-in XML parsers, most XML applications are Web applications.

Updategrams

Microsoft SQL Server 2000 has added XML functionality for Web applications to allow for XML inserts, deletes, and updates to batch operations. It allows changes to an XML document to be reflected in the SQL Server database. This is an add-on to SQL Server and is available in the XML for SQL Web Release 1, Beta 1, at the time of this writing. The actual syntax and use are subject to change, since the product is still in beta. It uses an XDR schema to map the XML to the database, which allows hierarchical updates to multiple tables. It uses optimistic concurrency. You can even use parameters for updategrams.

Downloading SQL Server 2000 XML Updategrams Support

To implement updategrams, you must download and install the XML for SQL Web Release. At this writing, you can download the software at the following URL:

http://msdn.microsoft.com/downloads/default.asp?URL=/code/sample.asp?url=/msdn-files/027/000/625/msdncompositedoc.xml

This download includes updategram support and the ability to load large XML documents into SQL Server tables using bulk load. To install updategram support, execute the file downloaded from the URL. By default, the filename is XML FOR SQL.EXE. You will get a dialog box asking you if you want to install the software. Click the Yes button to continue, and follow the instructions for installing. For additional help on using XML for SQL, you can access help files located in the following directory by default:

```
C:\Program Files\XML for SQL\Documents
```

Understanding Updategrams

Updategrams use an XDR schema in a manner similar to the way XPath uses one. The schema tells SQL Server how the database data map to the XML document. Some new syntax is added to XML documents to allow the updategrams to work. For these operations, three new elements are used: sql:before, sql:after, and sql:sync. These elements are used for inserts, deletes, and updates. To perform an insert operation, you

use only the sql:after element. To perform a delete operation, use only the sql:before element. To perform updates, both the sql:before and the sql:after elements are used.

Let's take a look at an example of an updategram. The sql:before element contains the current values in the XML document. The sql:after element contains the values to which you want to change. To change from an update operation to an insert or delete operation, simply remove the sql:before element for an insert operation or the sql:after element for a delete operation.

```
<root xmlns:sql="urn:schemas-microsoft-com:xml-updategram">
<sql:sync mapping-schema="SuppliersSchema.xml">
    <sql:before>
        <Suppliers SuppliersID="234" CompanyName="Previous Name"
                   Phone="(111)111-1111" sql:id="1">
    </sql:before>
    <sql:after>
        <Suppliers SuppliersID="234" CompanyName="New Name"
                   Phone="(222)222-2222" sql:id="1">
    </sql:after>
</sql:sync>
</root>
```

To execute the preceding updategram, you can execute the query directly from the URL or make a template file. If you saved the updategram to a template file named SuppliersUpdategram.xml, you would execute it like any other template file, as shown here:

```
http://localhost/XMLDemo/Queries/SuppliersUpdategram.xml
```

T-SQL OPENXML Statement

We have seen how we can retrieve data from SQL Server as an XML document. How do we work with XML documents using T-SQL in SQL Server? Microsoft SQL Server 2000 provides the OPENXML function to be used in T-SQL to create usable XML documents in SQL Server. OPENXML creates a rowset view of an XML document in memory. It allows you to access an XML document as though it were a relational rowset. This function can be used in SELECT and SELECT INTO statements. The syntax for the OPENXML function is as follows, with the arguments listed in Table 11.13 :

```
OPENXML(idoc int [in],rowpattern nvarchar[in],[flags byte[in]])
[WITH (SchemaDeclaration | TableName)]
```

Table 11.13: OPENXML Function Arguments

Argument	Description
Idoc	The document handle of the internal representation of an XML document created by sp_xml_preparedocument.
Rowpattern	The XPath pattern used to identify the nodes to be processed as rows.

Table 11.13: OPENXML Function Arguments	
Argument	**Description**
Flags	Sets the mapping that should be used between the XML data and the relational rowset.
SchemaDeclaration	The schema definition of the form.
TableName	The table name when a schema declaration is not used.
WITH clause	Provides a rowset format using either a schema declaration or specifying an existing table. If the optional WITH clause is not specified, the results are returned in an edge table format.

sp_xml_preparedocument and sp_xml_removedocument

The sp_xml_preparedocument stored procedure is used to create an internal representation of an XML document by parsing the XML document. It uses the MSXML parser. It returns a handle to access the rowset representation of the document. This rowset representation is stored in the SQL Server internal cache. The stored procedure returns 0 if it is successful or >0 if it fails. The syntax for the stored procedure follows, with the arguments in listed in Table 11.14 :

```
sp_xml_preparedocument

hdoc

OUTPUT [,

xmltext

][,

xpath_namespaces

]
```

Table 11.14: sp_xml_preparedocument Stored Procedure Arguments	
Arguments	**Description**
Hdoc	An integer that is the handle to the newly created document.
xmltext	The XML document as a text (*char* , *nchar* , *varchar* , *text* , *ntext*) parameter. If it is NULL, an internal representation of an empty XML document is created.
xpath_namespaces	The namespaces that are used in XPath expressions in OPENXML. The default value is <root xmlns:mp= ”urn:schemas-microsoft-com:xml-metaprop”>.

When you are finished using OPENXML, you need to remove the rowset representation from memory. The sp_xml_removedocument stored procedure removes it from memory

and makes the handle invalid. The syntax for this stored procedure follows. It takes an integer argument that is the handle created by the sp_xml_preparedocument stored procedure. It returns 0 if successful or >0 if it fails. If you pass it an invalid handle, it will return an error:

```
sp_xml_removedocument

hdoc
```

Let's take a look at an example of how OPENXML is used in T-SQL to insert a record into the database from an XML document:

1. Open the SQL Query Analyzer and connect to the Northwind database.
2. Let's enter the T-SQL necessary to insert the XML data. The first step is to call sp_xml_preparedocument to parse the XML document and get a handle to use with the OPENXML function. Enter the following text into the Query window:

```
3.        DECLARE @hDoc int
4.        EXEC sp_xml_preparedocument @hDoc OUTPUT,
5.             N'<ROOT>
6.                  <Suppliers CompanyName="Answermenow"
   ContactName= "Cameron"
7.                             ContactTitle="Engineer" Address=
   "123 Some St"
8.                             City="Melbourne" Region="FL"
   PostalCode= "32904"
9.                             Country="USA" Phone="(321)555-1212"
10.                            Fax="(321)555-1212"
11.                            Homepage="www.answermenow.com"/>
       </ROOT>'
```

12. Now we will insert the row into the Suppliers table and then remove the internal representation of the rowset. Enter the following text in the bottom of the Query window:

```
13.       — Use OPENXML to provide rowset
14.       INSERT Suppliers
15.       SELECT *
16.       FROM OPENXML(@hDoc, N'/ROOT/Suppliers')
17.            WITH Suppliers
   EXEC sp_xml_removedocument @hDoc
```

18. Let's execute the query. Click the Execute Query button (the green arrow) or press f5 to execute the query. Alternatively, from the Query menu, select Execute. Your window will look similar to Figure 11.10 .

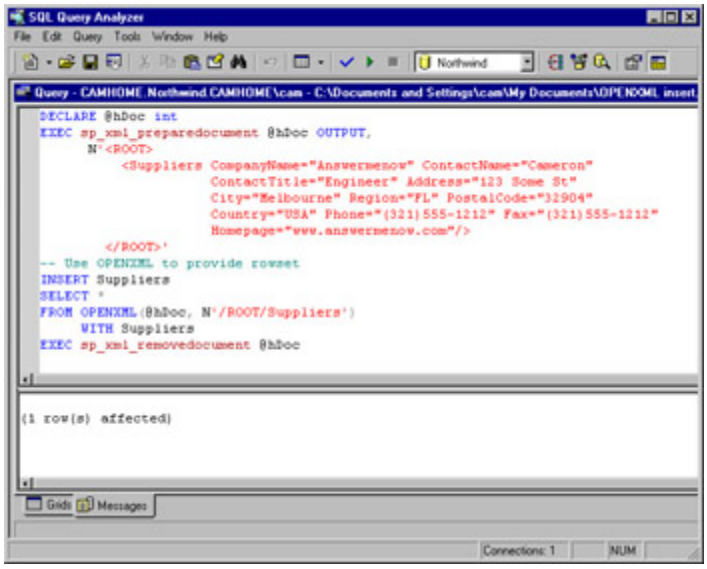


Figure 11.10: Executing the OPENXML function.

19. Let's verify that the record was inserted. Open a new query window by clicking the New Query button and pressing Ctrl-N , or from the File menu, select New.
20. Enter the following SQL statement in the Query window:

```
21.       SELECT * FROM Suppliers
   WHERE CompanyName='Answermenow'
```

22. Let's execute the query. Click the Execute Query button (the green arrow) and press f5 to execute the query, or from the Query menu, select Execute. Your window should look similar to Figure 11.11 .

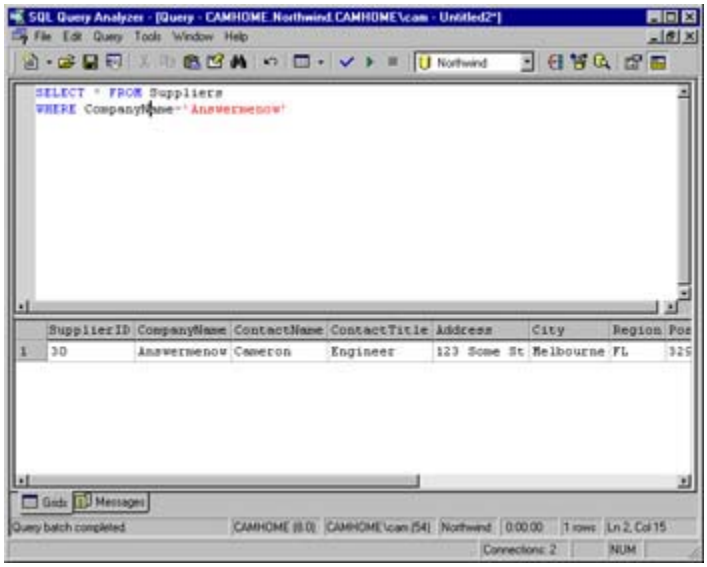


Figure 11.11: Verifying that the record was inserted.

ActiveX Data Objects

The XML features in Microsoft SQL Server 2000 can be used by applications through the Microsoft OLE DB Provider for SQL Server, which allows applications to use ADO to access these features via a data access model with which many developers are already familiar. With ADO, applications can use template queries, XML views, and the OPENXML function. You can also open and save XML files and load them into an ADO Recordset object. This is accomplished using the Recordset object's Open and Save methods.

The following is an example of saving a recordset as an XML file and loading a recordset with data from an XML file. This feature is available in ADO 2.1 and higher. To save the recordset as an XML file, you must use the adPersistXML option. One caveat of this is that ADO expects a specific format when reading an XML file. It is not flexible:

```
rs.Save "c:\XML Files\Suppliers.xml",adPersistXML


rs.Open " c:\XML Files\Suppliers.xml",,,,adCmdFile
```

XML Support in ADO 2.6

With the release of ADO 2.6, some properties have been added that are specific to using XML with SQL Server 2000. The OLE DB Provider for SQL Server 2000 exposes the new properties. These properties are used to specify XDR schemas for XPath queries and to specify XSL files. Table 11.15 lists these new features and their descriptions.

Table 11.15: New OLE DB Properties for XML in SQL Server 2000

Property Name	**Description**
Base Path	Specifies a file path or URL for resolving relative paths in a template.
Content Type	Returns the output content type of an XML transmission.
Mapping Schema	Specifies a filename or URL that points to the mapping schema used for XPath commands.
SS STREAM FLAGS	Specifies how an application maps schemas, XSL files, and templates.
XML Root	Provides a root tag for query results to return a well-formed document.
XSL	Specifies an XSL filename or URL applied to the result of a query.
Output Encoding	Specifies the encoding to use in the stream set or returned by the Execute method.
Output Stream	Specifies the stream containing the results returned by the Execute method.

Let's take a look at an example of retrieving XML data from SQL Server 2000 and pushing it down to the client for processing. When using XML from SQL Server in ADO, you must use the Stream object. You cannot use the Recordset object. The following is a sample ASP page. Some comments are added throughout the code as explanations:

```
<%@ LANGUAGE = VBScript %>
<% Option Explicit   %>


<HTML>
<HEAD>
<TITLE>FOR XML Query Example</TITLE>


<!— #include file="adovbs.inc" —>
<%

   Dim adoConn
   Set adoConn = Server.CreateObject("ADODB.Connection")

   Dim sConn
   sConn = "Provider=SQLOLEDB;Data Source=camhome;" & _
              Initial Catalog=Northwind;User ID=SA;Password=;"

   adoConn.ConnectionString = sConn
   adoConn.CursorLocation = adUseClient




   adoConn.Open

   Dim adoCmd
   Set adoCmd = Server.CreateObject("ADODB.Command")
   Set adoCmd.ActiveConnection = adoConn
```

The following is where the query for the XML document is specified. In this example, we are requesting the first five rows from the Suppliers table. Also notice that we are using the ADODB.Stream object rather than a recordset:

```
  Dim sQuery
  sQuery = "<ROOT xmlns:sql='urn:schemas-microsoft-com:xml-sql'>" & _
           "<sql:query>SELECT TOP 5 * FROM Suppliers " & _
                       "FOR XML AUTO</sql:query> " & _
           "</ROOT>"

  Dim adoStreamQuery
  Set adoStreamQuery = Server.CreateObject("ADODB.Stream")
  adoStreamQuery.Open
  adoStreamQuery.WriteText sQuery, adWriteChar
  adoStreamQuery.Position = 0
```

```
  adoCmd.CommandStream = adoStreamQuery
  adoCmd.Dialect = "{5D531CB2-E6Ed-11D2-B252-00C04F681B71}"

  adoCmd.Properties("Output Stream") = Response
  Response.write "<XML ID='XMLDemo'>"
  adoCmd.Execute , , 1024
  Response.write "</XML>"
%>
```

In the following code, the XML document object is created for parsing the XML data:

```
<SCRIPT language="VBScript" For="window" Event="onload">
   Dim xmlDoc
   Set xmlDoc = XMLDemo.XMLDocument




   xmlDoc.resolveExternals=false
   xmlDoc.async=false

   If xmlDoc.parseError.Reason <> "" then
      Msgbox "parseError.Reason = " & xmlDoc.parseError.Reason
   End If
```

Now we will step through the XML document nodes and display the CompanyName attributes returned from SQL Server 2000. The root variable is set to the XML document elements object, and then we will use a FOR EACH clause to step through all the child nodes. The getAttribute method is used to retrieve the actual value:

```
  Dim root, child
  Set root = xmlDoc.documentElement
  For each child in root.childNodes
     dim OutputXML
     OutputXML = document.all("log").innerHTML
     document.all("log").innerHTML = OutputXML & "<LI>" &
                                     child.getAttribute("CompanyName") &
"</LI>"
  Next
</SCRIPT>


</HEAD>
<BODY>
   <H3>Client-side processing of XML Document</H3>
   <UL id=log>
   </UL>
</BODY>
</HTML>
```

Summary

In this chapter, we examined the new XML features that have been incorporated in SQL Server 2000. We started out with an overview of XML and the new XML support added in SQL Server 2000. XML allows for data to be exchanged over HTTP in a text-based format. XML is also spreading to solutions besides the Internet. We saw how an XML document can be well formed and valid. XML data can be transformed using XSL for presentation of data or for formatting of data. The XPath language can be used to query XML data. In order to use XPath, you must map it to an XDR schema. An XDR schema allows you to provide an XML view.

One of the most intriguing new features in SQL Server 2000 is the ability to query a database from a URL. This feature allows data to be retrieved through HTTP without having to configure your network architecture. We stepped through the process of configuring IIS to allow URL queries by setting up a virtual directory and virtual names. From the URL, you can specify SQL statements, execute XML query templates, or execute XPath queries. You can even pass parameters to make your queries more dynamic.

The XPath language allows you to query XML documents for subsets of data. However, XPath queries must be executed against XDR schemas. Not all the W3C's XPath specification is currently supported by SQL Server 2000. Be sure to review the section for the limitations prior to creating XPath queries. Also remember that different XPath operators treat data types differently and can cause and unexpected subtle changes to results.

SQL Server 2000 has added the FOR XML SQL clause to allow you to retrieve data in XML. You can choose from three modes: RAW, AUTO, and EXPLICIT. RAW is the default mode. You can use the EXPLICIT mode to completely specify the format and hierarchy of the returned XML document. You must use the FOR XML clause in a SELECT statement. It cannot be used in UPDATE, INSERT or DELETE statements.

Using XDR schemas, you can create XML views. These are similar to views you create in SQL Server. An XDR schema limits the data that can be queried against. It is a mapping between the XML document to be retrieved and the underlying database schema. XPath queries are run against XDR schemas. You can also transform between the database data types and XML data types, as desired. Microsoft has also developed a tool called the SQL Server XML View Mapper to assist you in creating XDR schemas without having to learn the XDR language.

Finally, we discussed a couple of ways to update XML data between the client and the server. Updategrams allow you to update, insert, and delete data. They allow you to pass an XML document that contains only the data that need to changed. This is more efficient than retransmitting the entire document and then parsing for changes. This functionality is not part of the first release of SQL Server 2000 and must be downloaded and installed separately. You can also use the OPENXML function in T-SQL to update a SQL Server

database from an XML document. This function converts an XML document to a rowset in memory, allowing you to access the XML document as though it were a relational rowset.

FAQs

Q. I would like to pass data down to a client's Web browser in an XML document. I want the client to be able to view these data in a table or a graph. How can I do this without resending the entire data set to switch between these presentations?

A. You can pass the XML document down to the browser along with the XSL files. To change between the presentations, you don't need to resend the data, simply change the XSL file that is formatting the data.

Q. I want to be able to take updates to an XML document from the client and have those changes reflected in my SQL Server 2000 database. How can I accomplish this goal?

A. This task can be accomplished with either updategrams or using the T-SQL OPENXML function.

Q. I want to create a Web page that allows the user to perform XPath queries against a database. However, I want to limit the data that the user can query against based on the user level. How can this be done?

A. You can use XDR schemas to limit the data to query against. Create an XDR schema for each of the user levels you have. When the user logs in, you allow him or her only to query against the XDR schema file for that user level.

Chapter 12: Database Replication Techniques and Configuration

Introduction

At the heart of SQL Server replication are the tasks of making data available where they can best be used in your organization and optimizing application performance. Managing distributed data can be daunting without the services available in SQL Server replication. SQL Server 2000 offers several replication methods through numerous enhancements in the management, distribution, and implementation of replication over previous versions of SQL Server.

New to SQL Server replication is support for queued updating subscribers. Organizations implementing remote and often disconnected client applications will find immediate advantages in SQL Server 2000's replication feature. Disconnected applications can now update local copies of data and perform merge replication when they are connected to the server network. Several enhancements to merge and transactional replication increase the performance and stability of replication solutions.

For organizations new to replication, additional wizards and a simplified setup help get replication up and running with minimal effort. Through additional support for Windows 2000 Active Directory, locating and using publications in the enterprise are transparent to your server configurations. Publications are made available with Active Directory search and browse capabilities. Monitoring SQL Server replication and dealing with associated problems are accomplished using the Replication Monitor. Improvements to error handling and conflict resolvers make SQL Server 2000 replication the most reliable and manageable release yet.

If your organization has implemented a previous version of SQL Server replication or has struggled with distributing data throughout the organization and remote clients, the question of whether or not to upgrade to SQL Server 2000 has been answered for you. This chapter reviews the architecture, enhancements, and new features of SQL Server replication. You will enable and configure replication for your SQL Server and locate and subscribe to publications. The review of monitoring, backup, and performance considerations will help you keep your replication environment performing at optimal levels.

SQL Server Replication Architecture

SQL Server 2000 database replication can be accomplished using three different methods or types, but the basic model and components are constant. All replication scenarios consist of a publisher, a distributor, and a subscriber, as depicted in Figure 12.1 . These each consist of a set of service agents that can be run on one or more servers and that can have differing roles, depending on your chosen replication method.

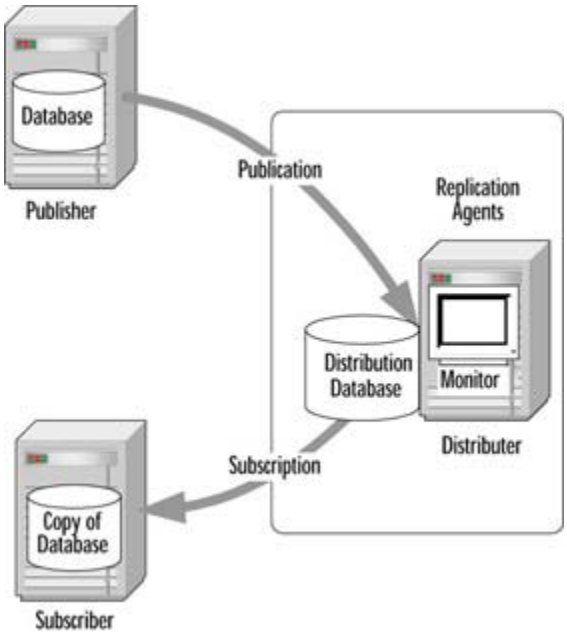


Figure 12.1: SQL Server 2000 replication roles.

Publisher

The server that contains the actual database or databases to be replicated is referred to as the *Publisher* . The Publisher also keeps track of the specific modifications that have been made to the replicated data. Any server running SQL Server 6.5 or higher (depending on replication method) or one of several heterogeneous databases such as Oracle or DB2 can publish data for replication with SQL Server 2000.

Subscriber

The recipient of a Publisher's replicated data is called the *Subscriber* . Each Publisher can have one or multiple subscribers. A subscriber can be the same server as the Publisher, or it can be any computer running a compatible version of SQL Server (6.0 or higher), including SQL Server 2000 Windows CE Edition, or a compatible heterogeneous database. Subscribers can be configured to replicate to a local Publisher or to a Publisher anywhere in the world via the Internet.

Distributor

Information about the replicated data is stored in a database on the *Distributor* . The distribution database contains replication history data, metadata, and/or transactions, depending on the type of replication you implement. The Distributor can be located on the same server as the Publisher (a *Local Distributor*), or it can be on a separate server (a *Remote Distributor*).

Publication

The complete data set made available by the Publisher is called a *publication* . A publication consists of one or more articles.

Article

A publication can contain any number of database objects such as tables, stored procedures, or views. It also might contain filtered data from tables. Each of these individual published items is referred to as an *article* . Specific replication properties can be set for each article, including the type of logging, the source and destination objects, any data filters, and the schema generation scripts or options.

Subscription

The request for a publication to be sent to a Subscriber is called the *subscription* . Subscriptions contain information about the publication and the replication schedule. A subscription can be created at the request of either the Publisher or the Subscriber. There are two basic types of subscriptions:

- A *push subscription* is created at the Publisher/Distributor, and the synchronizing agent is run at the Distributor.
- A *pull subscription* is created at the Subscriber, and the synchronizing agent usually runs at the Subscriber.

Additionally, a publication can be set up to allow *anonymous subscriptions* , which are a type of pull subscription that does not store information at the Publisher.

SQL Server Agent

The SQL Server Agent acts as the “ringmaster” for all the replication processes in that it is used for scheduling and managing all the other replication agents. The SQL Server Agent starts each of the replication jobs running on its set schedule, keeps a historical log of replication activities, and notifies the administrator of replication process failures.

Replication Agents

SQL Server 2000 Replication comprises a group of agents that each have a task relating to their unique portion of the replication process. Not all agents are used for all types of replication. The following points summarize the roles of the individual agents:

- **Snapshot Agent** Runs on the Distributor. Prepares the initial schema and data files of a publication into a package called a *snapshot* , then stores the snapshot files in either the default location or the location specified during publication creation (which might be a shared folder or an FTP site). *Used with all replication types* .
- **Distribution Agent** Runs at the Distributor if it is a push subscription or at the Subscriber for a pull subscription. Physically distributes the snapshot jobs and transactions from the distribution database to the Subscriber. Used with both snapshot and transactional replication.
- **Merge Agent** Runs at the Distributor if it is a push subscription or at the Subscriber for a pull subscription. A separate instance of the merge agent is run for each merge subscription. Connects to both the Publisher and the Subscriber, moving changed data from one and applying them to the other. Merge types can be bidirectional, upload, or download, depending on how the agent is configured. *Used with merge replication only.*
- **Log Reader Agent** Runs on the Distributor and connects to the Publisher. A separate instance of this agent runs for each publication database. Copies transactions from the Publisher's transaction log to the distribution database. *Used with transactional replication only.*
- **Queue Reader Agent** Runs on the Distributor. One instance of this agent services all Publishers and publications on Distributor. It reads queued transaction messages and applies them to each publication as needed. *Used with both snapshot and transactional replication, if the Queued Updating option is enabled (or configured as a fail-over for immediate updating).*
- **Miscellaneous Agents** History, distribution database, and expired subscription cleanup, plus subscription reinitialization and replication agents checkup. Each can be scheduled or re-scheduled from the from the Replication Monitor folder.

New Replication Features in SQL Server

Replication has undergone significant changes in SQL Server 2000, making it much more administrator-friendly and powerful. There are also some impressive new features, including:

- **Queued updating** Allows Subscribers to modify data while it is physically disconnected from the Publisher. Modification transactions are queued up, and changes are applied to the Publisher when the connection is restored. Changes are made asynchronously, but conflict resolution options are presented if are multiple modifications are made to the same data by different Subscribers simultaneously.
- **Transformable subscriptions** Allow each Subscriber to request a customized version of the published data based on its own unique requirements.
- **Replication support for new SQL Server 2000 features** These features include indexed views (which can be published as tables), user-defined functions, new data types, and multiple SQL Server instances.
- **ActiveX Snapshot Control** Allows developers to create snapshots more easily from within applications.

- **The ability to browse for and subscribe to replication publications** using Windows 2000 Active Directory.
- **Support for heterogeneous data sources** (Microsoft Access, Oracle, DB2, etc.) via COM interfaces and replication with SQL Server 2000 CE Edition.
- **Schema change replication** This is a particularly useful feature in a development environment in which you might have several copies of a database on different servers and need to make sure that when you add or drop schema elements on one database, the change is synchronized to all other servers.
- **On-demand, pre-snapshot, or post-snapshot execution of scripts on all subscribers** Useful for actions such as verification, cleanup, or archiving.

Replication Compatibility

Since simultaneously upgrading all your clients, applications, and servers to SQL Server 2000 might not be feasible, you'll probably want to replicate data between your older servers and those that have been upgraded. You might also want to replicate data from heterogeneous data sources within your company or from your customers or vendors. Fortunately, these options are possible with SQL Server 2000 replication.

Previous Versions of SQL Server

Replication in SQL Server 2000 is backward compatible with previous versions; however, its new features and enhancements will not function when replicating with older versions. Functionality is limited to the oldest version of SQL Server in your replication schema. For this reason, only certain combinations of versions will work as Publishers, Distributors, and Subscribers for each type of replication:

- Transactional and snapshot replication:
  - The Publisher can be running SQL Server version 6.5, 7.0, or 2000.
  - The Distributor must be the same version (or a later version) as the Publisher (version 6.5 or higher).
  - The Subscriber can be running any version 6.0 or above. If the subscriber is version 6.0, it must be configured as an ODBC data source; it cannot be configured as a native SQL Server Subscriber.
- Merge replication:
  - The Publisher can run on SQL Server version 7.0 or 2000.
  - The Distributor must be SQL Server 2000.
  - The Subscriber must be version 7.0 if Publisher is 7.0, or it can be 7.0 to 2000 if the Publisher is running 2000.

Note To use replication with older versions of SQL Server, ensure that you have met these minimum service pack requirements:

- SQL Server 6.5 must be running Service Pack 4 or higher.
- SQL Server 7.0 must be running Service Pack 1 or higher.

Heterogeneous Publishers and Subscribers

Complete standardization is a rarity in large corporate infrastructures, so if your company needs to share SQL Server data with applications on different database platforms, SQL Server 2000 can help. SQL Server 2000 allows you to publish data to most non-SQL Server data sources, as long as the Subscriber provides a 32-bit ODBC or OLE DB data access driver. It can also subscribe to data when the Publisher is one of these heterogeneous data sources: Microsoft Access, Oracle, DB2 (Universal, MVS, and

AS400), and Microsoft Exchange Server. In addition, see the sections on publishing to the Internet and to SQL Server 2000 CE Edition later in this chapter.

Heterogeneous Subscribers

Data published to a heterogeneous source are most simply handled using OLE DB or ODBC and then creating a push subscription from the Publisher to the OLE DB or ODBC Subscriber. Alternatively, you can create a publication and then an application with an embedded distribution control, which serves to implement the pull subscription from the Subscriber to the Publisher.

Note For ODBC Subscribers, the subscribing data source has no administrative capabilities in regard to the replication being performed.

To enable heterogeneous Subscribers, select the option labeled "Heterogeneous data sources," such as Oracle or Microsoft Access; devices running SQL Server CE; or servers running earlier versions of SQL Server on the Specify Subscriber Types screen of the Create Publication Wizard.

Warning If you do not enable heterogeneous Subscribers when you create the subscription database in the Create Publication Wizard, the schema will be published to the Subscriber but the data will not be, and you will not receive an error message.

Heterogeneous Publishers

SQL Server 2000 can subscribe to data replicated from Oracle, DB2, Access, and other data sources, using the snapshot or transactional replication methods. To enable SQL Server applications to subscribe to publications from heterogeneous data sources, configure SQL Server with third-party software or use applications built with SQL-DMO and the Replication Distributor Interface.

Designing for Database Replication

Self-queries about your organization's objectives in distributing data ensure that replication is the correct solution for the problems you need to solve or the right direction for the goal you hope to attain. Queries about your organization's needs regarding data distribution help you determine the type of replication needed, where it is needed, and how often it is needed.

Replication Requirements

To successfully utilize SQL Server Replication technology across your organization, whichever specific method you choose, you should take into account these basic requirements:

- Data physical location and site autonomy
- Data modification consistency
- Network connection bandwidth and availability
- Application and database design for conflict prevention

Data Location

Data location is a major factor that you must take into account when deciding whether to use replication and what type of replication to use. You might need to consider some of the following questions:

- Where will the master copy of each segment of your database reside?
- Which site in your organization will "own" the data?
- What security restrictions are placed on data access at the locations to be replicated?
- Can data be partitioned logically so that sites can modify their own data without causing conflicts with updates made at other sites?

Data Modification

To help you determine the types of replication to use, the replication options to use, and when to schedule updates, consider the following questions:

- Will Subscribers be read-only, or will they need to update the data? If so, how much, and how often?
- If multiple Subscribers need to update the same sets of data, what kinds of conflicts will occur? How will those conflicts be handled?
- How quickly will updates need to be applied to keep them in sync with all Subscribers?

Connection Bandwidth and Availability

Another big question to consider is, will Subscribers have continuous, reliable connections to the publication database, or will they be disconnected for periods of time?

Your network's speed and availability can have either a positive or a negative effect on replication. Replication can also have an adverse effect on your network. Issues that can affect the performance of your network are:

- Volume of data flowing over the network
- How many Subscribers there are to a particular Publisher
- Speed and overall reliability of the LAN or WAN link

Profiles for the agents involved in replication can be customized when you are replicating over a slow link. This capability allows you to configure behavior such as batch size, polling interval, time-out period, or amount of memory available for buffering data.

When applying the initial snapshot, network speed is often the most important issue. Even if the volume of incremental data changes will be low, the volume of data initially sent to the Subscribers might be quite high. Solutions to data volume fluctuations include:

- Transferring the initial snapshot using a CD-ROM or tape device
- Compressing the snapshot files
- Performing pretransformations on published data that discard some data at the Distributor

Application and Database Design Considerations

Database design ultimately determines the complexity and processing resource requirements of queries used by merge replication, which in turn affects merge performance. Improving merge performance requires some structural changes to the database. Columns or tables can be added to support dynamic partitioning logic more efficiently while ensuring that the columns used in the filtering expressions can take advantage of indexes. Using indexes on all filtering columns might be counter-productive in terms of index maintenance (especially if the index is not used by the query optimizer) because the data are not very unique or the expression cannot use indexes. Sometimes changing the filtering expressions allows an existing index to be used where it was not before.

Additional design considerations that you must make when publishing data for updating by multiple Subscribers include:

- **Minimize potential conflicts** Use the NOT FOR REPLICATION flag and specialized data types such as *uniqueidentifier* and *timestamp* for conflict resolution during replication
- **Manage table RowGuid and identity values** Specifying identity range management helps control data modifications to various Subscribers during merge replication, snapshot, or transactional replication with updateable subscriptions.

Minimize Potential Conflicts

Conflicts can arise when certain data types or database properties that are being replicated from one server to another contain unique key fields that are not unique across servers. These conflicts can be resolved by the servers during replication using specialized data types such as *timestamp* or *uniqueidentifier* .

SQL Server 2000 *timestamp* data indicate the relative sequence of data modification within a database by use of database-specific incrementing binary numbers. *Note:* Timestamp *data are unrelated to both chronological time and calendar date. Timestamp* data is no longer used for conflict detection. A *uniqueidentifier* data type column is used instead.

The following are implications of *timestamp* data processing:

- For conflicts to occur with row-level tracking, the same row must be updated at both replicas.
- For conflicts to occur with column-level tracking, the same column within the same row must be updated at both replicas.

Note When the literal values for a *timestamp* column are replicated, its data type is changed to binary (8) on the Subscriber. During merge replication and queued updating, Subscriber articles containing a *timestamp* column are replicated, but the literal *timestamp* values are not. *Timestamp* values are regenerated at the initial synchronization time, when the rows are applied at the Subscriber. Merge replication tracking ignores *timestamp* values. The queued updating Subscribers option for transactional replication uses only row-level tracking to detect conflicts.

Another way to minimize replication conflicts is to use the NOT FOR REPLICATION option when implementing check constraints or for ranges of identity values in a partitioned environment. This option is especially useful in transactional or merge replication when a published table is partitioned with rows from various sites.

Table RowGuid and Identity Values

Columns with the IDENTITY property are system-generated sequential integers that are often used as primary keys because they are unique when they are assigned to each row in a table. IDENTITY columns pose a bit of a problem during replication between servers because the sequential number used on each server has the potential to be accidentally duplicated. There are several ways to get around this problem:

- Allow SQL Server to automatically assign identity values based on predefined ranges for the Publisher and Subscribers during replication.
- Use a different range of IDENTITY columns for the Publisher and Subscribers and manually manage which IDENTITY seed values are assigned by which server.

- Use the UNIQUEIDENTIFIER column data type instead.
- Use a compound primary key from other meaningful data instead.
- Use the T-SQL statement: NOT FOR REPLICATION when creating IDENTITY columns. This tells SQL Server to use the existing value in the IDENTITY field from the publisher and not to assign a new one during replication.

Merge replication uses ROWGUIDCOL columns frequently during tracking and synchronization of changes made at the Publisher and the Subscribers. ROWGUIDCOL is a property you can assign to a column with the UNIQUEIDENTIFIER data type. UNIQUEIDENTIFIERS, or *globally unique identifiers (GUID)* , are unique to each server because each is a 128-bit integer generated from a unique combination of data. To create a GUID, SQL Server generates a binary value from the computer's network card identification number (its MAC address) and a unique number generated from the CPU clock. For this reason, using a GUID column as a primary key is a safe alternative to using an identity column to guarantee uniqueness within the database and across multiple servers.

Merge replication requires each published table to have a ROWGUIDCOL column. Creating a ROWGUIDCOL column on each table prior to generating the Initial Snapshot for a merge publication will help you avoid the significant time performance decrease that can occur while waiting for the Snapshot Agent to alter the tables for you. These columns can have any name (the Snapshot Agent uses ROWGUID as its default), but all have the following data type characteristics:

- The ROWGUIDCOL property must be assigned.
- The data type as UNIQUEIDENTIFIER with default set to the NEWID() function.
- There must be a unique index on the column.

NOT FOR REPLICATION is an option set when creating the table; it is activated when a replication agent connects to that table. The option forces SQL Server to maintain the original identity values on rows replicated to Subscribers but continues to increment the identity value on rows added through normal user activity.

Note If you are using transactional replication with the immediate-updating Subscribers option, do not use NOT FOR REPLICATION on the Subscriber. Instead, create the IDENTITY property at the Publisher only, and have the Subscriber use just the base data type (for example, *integer*). Then, the next identity value is always generated at the Publisher.

Setting the NOT FOR REPLICATION Flag on Check Constraints

If your database was not designed initially with replication in mind, you do not have the NOT FOR REPLICATION flag already set for all your table constraints. Since there is no ALTER CONSTRAINT command, the only way to set the flag is to drop and then add each constraint individually with the NOT FOR REPLICATION flag specified. If you

don’t have many constraints, this might not be a problem, but if you have a large number to change, you can use SQL-DMO to set the ExcludeReplication property value for each constraint programmatically.

The following is an Active Server Pages VBScript example that will perform this task for every constraint in a given database. Save this script to your Web server and execute it to update your database:

```
ScriptName: Noreplconstraints.asp
<%@ Language=VBScript %>
<%
Server.ScriptTimeout=6000
if request("dbname")<>"" and request("server")<>"" then
Dim oSQLServer, oDatabase, oTable, oCheck, oKey, oNewKey
Dim bConnected
dim strServer, strDBName, strKey, strReferencedKey, strName
dim strReferencedTable, strMessage
dim aKey(100)




set oSQLServer = Server.CreateObject("SQLDMO.SQLServer")
set oTable = Server.CreateObject("SQLDMO.Table")
set oCheck = Server.CreateObject("SQLDMO.Check")
set oKey = Server.CreateObject("SQLDMO.Key")

strServer=request("server")
strUser=request("user")
strPW=request("pw")

'Connect to Database
oSQLServer.LoginTimeout = 30
bConnected = oSQLServer.Connect(strServer, strUser, strPW)

if bConnected=0 then
   strDBName = request("dbname")
   strMessage="List of Changed Items:<br>" & vbCRLF
   for each oTable in oSQLServer.databases(strDBName).tables
       for each oCheck in oTable.checks
           oCheck.ExcludeReplication=True
           strMessage = strMessage & "Check Constraint:" &
oCheck.Name & " on Table: " & oTable.Name & "<br>" & vbCRLF
       next
       x=0
       for each oKey in oTable.keys
           if oKey.Type=3 then
              x=x+1
              aKey(x)=oKey.name
           end if
       next
       for y = 1 to x
           strKey = aKey(y)
           set oKey = oTable.keys(strKey)
```

```
          strReferencedTable = oKey.ReferencedTable
          strReferencedKey = oKey.ReferencedKey
          strName = oKey.Name
          set oNewKey = Server.CreateObject("SQLDMO.Key")
          oNewKey.Name = strname
          oNewKey.Type = 3
          oNewKey.ReferencedTable = strReferencedTable
          For iColumn = 1 to oKey.KeyColumns.Count
              strKeyColumnName = oKey.KeyColumns(iColumn)
              oNewKey.KeyColumns.Add(strKeyColumnName)
          Next
          For iColumn = 1 to oKey.ReferencedColumns.Count
              strKeyColumnName = oKey.ReferencedColumns(iColumn)
              oNewKey.ReferencedColumns.Add(strKeyColumnName)
          Next
          oNewKey.ExcludeReplication=True
          oTable.keys.Remove(strName)
          oTable.keys.Add(oNewKey)
          strMessage = strMessage & "Foreign Key:" &
oNewKey.Name           & " on Table: " & oNewKey.ReferencedTable
& " Referencing: " & oNewKey.ReferencedKey & "<br>" & vbCRLF
            set oNewKey=nothing
        next
   next
   oSQLServer.DisConnect
else
   strMessage="Login Failed!"
end if
set oSQLServer = nothing
set oDatabase = nothing
else
   strMessage="Please Enter a Valid Database and Server Name"




end if
%>
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
</HEAD>
<BODY>
<center><h3>Type Database Server and Name and Click SUBMIT to
Exclude All Constraints from Replication</h3></center>
<center><h2>WARNING: This script takes a while to
run!</h2></center>
<form name=form1 action="noreplconstraints.asp" method="post">
<table>
<tr><td>SQL Server Name:</td>
<td><input name="server" size=20 value="<%=request("server")%>">
</td></tr>
<tr><td>Database Name:</td>
<td><input name="dbname" size=20 value="<%=request("dbname")%>">
```

```
</td></tr>
<tr><td>SQL Server User Name (sa):</td>
<td><input name="user" size=20 value="<%=request("user")%>"></tr>
<tr><td>Password:</td><td>
<input name="pw" size=20 type=password value="<%=request("pw")%>">
</tr>
<tr><td align=center colspan=2>
<INPUT type="submit" value="Submit" id=submit1 name=submit1>
<INPUT type="reset" value="Clear Form" id=reset1 name=reset1>
</td></tr>
</table>
<center><font color=darkblue><%=strMessage%></font></center>
</BODY>
</HTML>
```

Replication Methods in SQL Server

Three types of replication can be implemented in SQL Server 2000: snapshot replication, merge replication, and transactional replication. Depending on your application and data requirements, you might choose only one of these types or any combination of them.

Transactional Replication

Transactional replication (see Figure 12.2) uses an initial snapshot of the data applied to the Subscribers, then captures and propagates the individual transactions to Subscribers once data modifications have been made at the Publisher.

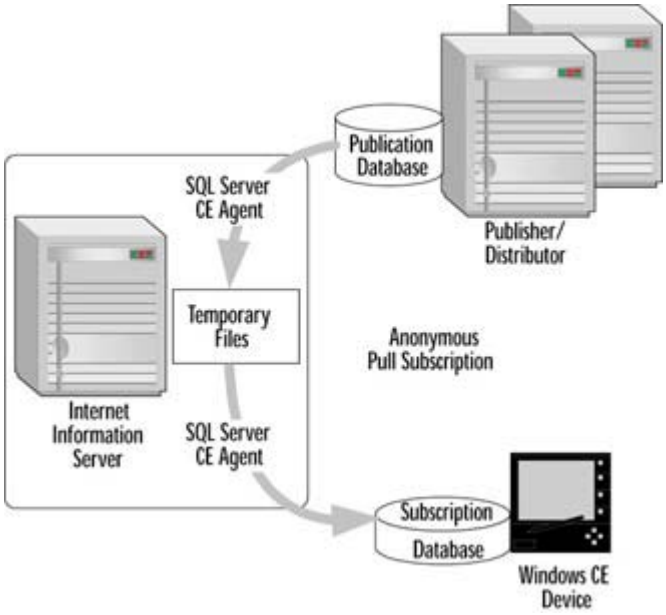


Figure 12.2: Transactional replication.

Transaction logs are used by transactional replication to capture each incremental change made to data in a published table. Microsoft SQL Server 2000 monitors INSERT, UPDATE, and DELETE statements and stores the changes in the distribution database before applying them to Subscribers in the occurring order.

Incremental changes made at the Publisher flow according to the Distribution Agent schedule, which can be set to continuous for minimal latency or to scheduled intervals to Subscribers. Update conflicts are avoided because all changes to the data must be made at the Publisher (when transactional replication is used without immediate updating or queued updating options). All Subscribers will end up achieving the same values as the Publisher.

The Snapshot Agent, the Log Reader Agent, and the Distribution Agent are responsible for implementing transactional replication. Snapshot files containing schema and data of published tables as well as database objects are prepared by the Snapshot Agent, which

then stores the files in the snapshot folder and records synchronization jobs in the Distributor distribution database. The Log Reader Agent reads the transaction logs for the publication database and copies the transactions marked for replication into the distribution database. The Distribution Agent applies to the Subscribers both the initial snapshot and the transactions from the distribution database .

In order to purge the data that are no longer required, SQL Server adds certain tasks to the SQL Server Agent at the Distributor when the distribution database is created. Once all transactions have been applied to the Subscribers, the Distribution Cleanup Agent removes transactions that were delivered to the distribution database.

Note Replicated transactions are kept in the distribution database for a defined period of time, known as the *retention period* . Setting a retention period ensures that information required to automatically recover a destination database is available within the distribution database.

Queued Updating Subscribers

A high-speed network connection to the Publisher ensures that Subscribers receive data changes in near real time. Transactional replication can provide very low latency to Subscribers; provided that the network link and adequate processing resources are available, latency of a few seconds can often be achieved. Subscribers receiving data using a push subscription usually receive changes from the Publisher within 1 minute, although they can also pull changes down on an as-needed basis.

In addition, the use of transactional replication by disconnected users can be very effective for read-only data. *Queued updating* allows Subscribers to modify data while physically disconnected from the Publisher. Modification transactions are queued, and changes are applied to the Publisher when the connection is restored. Changes are made asynchronously, but conflict resolution options are presented if multiple modifications are simultaneously made to the same data by different Subscribers.

Merge Replication

Merge replication (see Figure 12.3) distributes data from Publisher to Subscribers while allowing both the Publisher and Subscribers to make updates to the data. Updates can be made when connected or disconnected. Once reconnected, the system merges updates between sites.

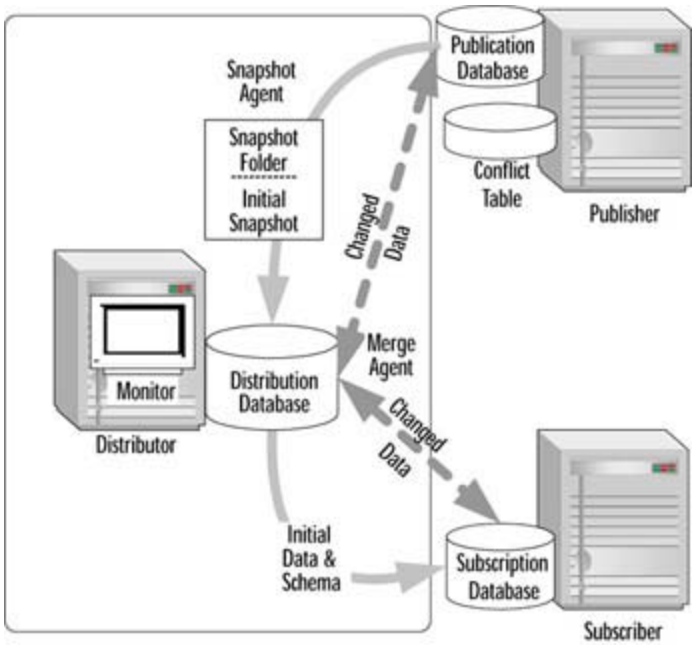


Figure 12.3: Merge replication.

Merge replication enables various sites to work autonomously and merge updates into a single, uniform result at a later time. Once the initial snapshot is applied to Subscribers, SQL Server 2000 tracks changes to published data at the Publisher and at the Subscribers. The data are synchronized between servers continuously, on demand or at scheduled intervals. It is possible for the same data to be simultaneously updated by the Publisher and/or more than one Subscriber, since updates are made at more than one server. Thus, conflicts can occur when updates are merged.

Note Both queued updating and merge replication allow offline updates at the Publisher and at Subscribers, although there are significant differences between the two methods.

Merge replication offers both default and custom selections for conflict resolution that can be defined as a merge publication is configured. Should a conflict occur, a resolver is invoked by the Merge Agent. The resolver determines the data that will be accepted and propagated to other sites.

The Snapshot Agent and the Merge Agent are responsible for implementing merge replication. Snapshot files containing schema and data of published tables as well as database objects are prepared by the Snapshot Agent, which then stores the files in the snapshot folder and records synchronization jobs in the Distributor distribution database. The Snapshot Agent also creates replication-specific stored procedures, triggers, and system tables.

Initial snapshot jobs held in the publication database tables are applied to the Subscriber by the Merge Agent. Incremental data changes that occurred at the Publisher or

Subscribers after the initial snapshot was created are also merged, and then conflicts are reconciled according to rules or a custom resolver you have created.

Implementing the Distributor locally (on the same server as the Publisher) is quite common because the Distributor role in merge replication is limited. The Distribution Agent is not used at all for merge replication, and the distribution database on the Distributor only stores history and miscellaneous information about merge replication; it does not store temporary copies of replicated data, as in transactional or snapshot replication.

SQL Server automatically adds tasks to SQL Server Agent, once the distribution database has been created, in order to purge the data no longer needed. Administrators should outline a periodic maintenance plan because these tasks help replication function effectively in a long-running environment. The cleanup tasks delete the initial snapshot for each publication when there are no more subscriptions to it and remove history information from the Msmerge_history table.

Snapshot Replication

Snapshot replication (see Figure 12.4) does not monitor updates to the data. Rather, it distributes data exactly as they appear at a specific point in time. Snapshot replication is best used for replicating data when the most current updated values are not a requirement, when data are used as read-only, or when the data change infrequently. Once synchronization occurs, the entire snapshot is generated and sent to Subscribers. Subscribers not updating data can be disconnected.

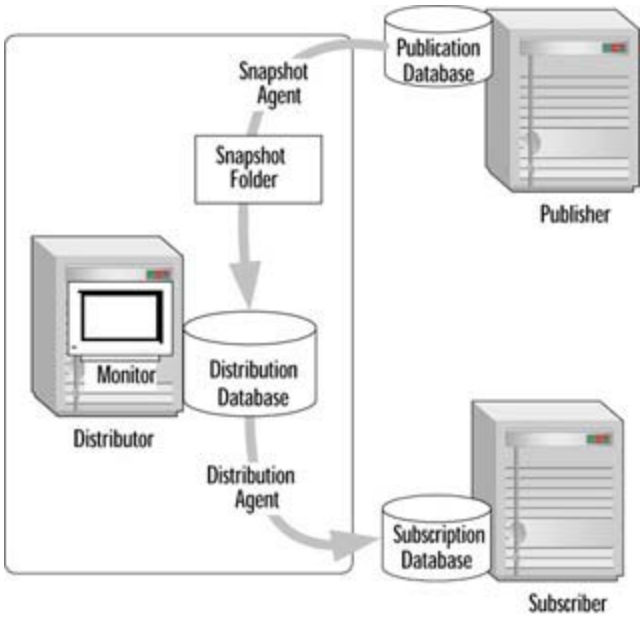


Figure 12.4: Snapshot replication.

Snapshot replication is particularly appropriate for the distribution of read-only data copies while providing the option to update data at the Subscriber. Transactional

consistency is maintained between the Publisher and Subscribers, when Subscribers only read data. When Subscribers must update data, however, transactional consistency can still be maintained between the Publisher and Subscriber because the data are propagated using a two-phase commit protocol (2PC). Snapshot replication does not require continuous monitoring of data changes on source servers and thus requires less constant processor overhead than transactional replication. In deciding whether or not snapshot replication is appropriate, you must consider the size of the entire data set and the frequency of changes to the data.

The Snapshot Agent and the Distribution Agent are responsible for implementing snapshot replication. Snapshot files containing schema and data of published tables as well as database objects are prepared by the Snapshot Agent, which then stores the files in the snapshot folder and records synchronization jobs in the Distributor distribution database. The snapshot folder is located on the Distributor by default, but you can specify an alternate location instead of or in addition to it. The Distribution Agent copies the snapshot from the snapshot folder to the destination tables at the Subscribers.

Additional tasks run on the Distributor for agent, transaction, and history cleanup to help replication perform well over the long term. When a snapshot has been applied to all Subscribers, replication cleanup deletes the associated initial snapshot (**.** BCP) file automatically. The initial snapshot is *not* deleted if anonymous subscriptions have been enabled for the publication or if the option to create a first snapshot immediately was used when snapshots were created. This feature ensures that a snapshot will always be available to new, anonymous subscribers.

Selecting a Replication Method

Each type of replication provides a different capability, depending on the application and the various levels of *atomicity, consistency, isolation, and durability (ACID)* properties of transactions and site autonomy. Generally, Publishers and Subscribers should be connected continuously in order for updates to be propagated to Subscribers; however, transactional replication maintains transactional consistency as well, although not as effectively as merge replication. Snapshot replication is preferable over transactional replication when data changes are substantial but infrequent. When publishing small tables that will be updated only at the Publisher, another option is to create new snapshots on daily intervals.

The same application can use multiple replication types and options. Some data in the application might require no updates at Subscribers; some sets of data might require updates infrequently; other sets of data might need to be updated daily at multiple servers.

Your specific requirements—based on distributed data factors, whether or not data need to be updated to the Subscriber, the replication environment, and the needs and requirements of the data to be replicated—will determine which type of replication you most need to implement.

Each type of replication begins with generating and applying the snapshot at the Subscriber, so it is important to understand snapshot replication in addition to any other type of replication and options you choose. The benefits of the types of replication are summarized here:

- Snapshot replication is best used when:
    - Data changes are infrequent, and publishing an entirely new copy to Subscribers makes more sense.
    - Servers do not need to be completely synchronized at any given time.
    - The database is small enough to copy it in its entirety on a regular schedule without degrading server and network performance.
- Transactional replication is best used when:
    - Incremental changes need to be sent to Subscribers immediately as they occur.
    - Transactions cannot be allowed to conflict or violate ACID rules.
    - Subscribers can connect reliably and/or frequently to the Publisher.
- Merge replication is best used when:
    - Subscribers might or might not be connected to the Publisher, and they need to be able to make updates offline.
    - More than one Subscriber needs to update the data, and the data need to be kept in sync with all the other Subscribers and the Publisher.
    - Conflicts are expected to be minimal when data are updated at multiple Subscriber sites; however, some violation of ACID rules can be tolerated.

Configuring SQL Server Replication

Replication configuration varies depending on the type of replication selected. In general, this process can be broken down into three steps:

1. Enable server publishing. Configure a Publisher and a Distributor.
2. Create publications. Define the specific data or subsets of data you want to replicate.
3. Add Subscribers. Push or pull Subscriptions, and configure your replication schedule and options.

Enabling Server Publishing

To enable publishing and configure your server as a Publisher and/or Distributor, use either the Create and Manage Publications option or the Configure Publishing, Subscribers, and Distribution option on the Replication task list in Enterprise Manager (see Figure 12.5). Running one of these options will invoke a wizard that walks you through step-by-step configuration of your server.

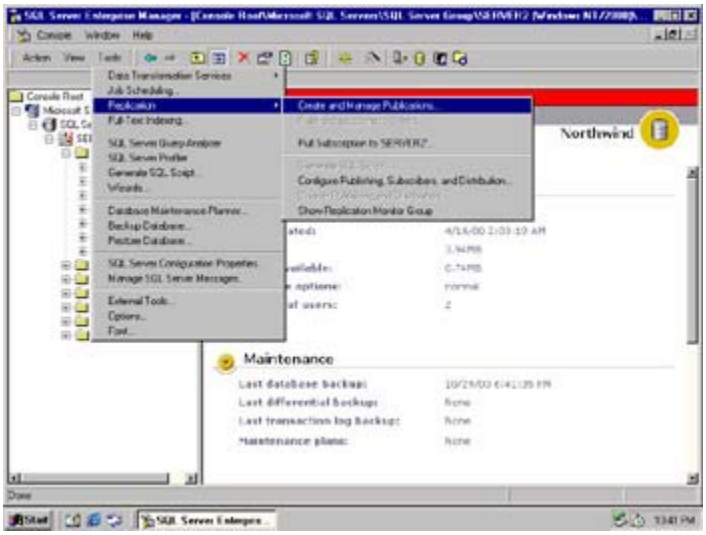


Figure 12.5: Configuring replication in Enterprise Manager.

Assigning a Distributor Server

When you set up a new Publisher, you are prompted to choose a Distributor for that Publisher (see Figure 12.6). Snapshot and transactional replication require a Distributor as a main participant in the process. The Distributor stores transactions temporarily during transactional replication, and most of the replication agents are run on it. For transactional replication, best performance is achieved if there is a Remote Distributor on a separate computer from the Publisher. Merge replication uses a Distributor in a much more limited role (mostly for monitoring and storage of log data), so a Local Distributor can be used.

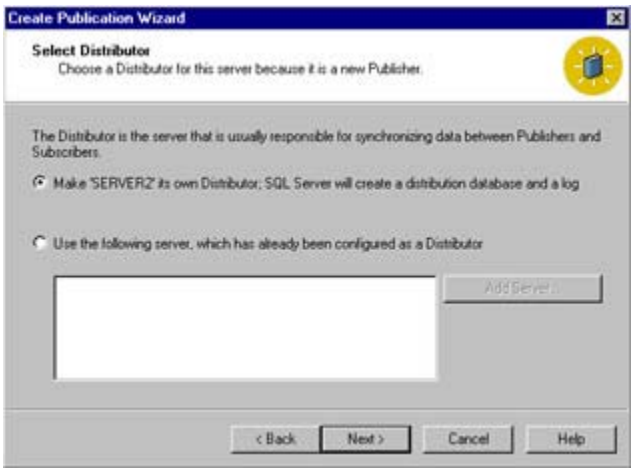


Figure 12.6: The Select Distributor screen.

Creating Publications

Use the Create Publication Wizard in Enterprise Manager to define publication databases and filters and set publication properties. To start the Wizard, click the Create Publication button from within the Create and Manage Publications screen.

Create Publication Wizard

The following is an example of the steps you will go through for setting up a snapshot publication to the Northwind database using the Create Publication Wizard:

1. Select the Northwind database from the Choose Publication Database screen, then click Next.

   Note On the Welcome screen, be sure to check the Show Advanced Options in this Wizard box if you need to perform any advanced functions such as publishing to non-SQL Server clients.

2. In the Select Publication Type screen, select the publication replication method that best suits the needs of your application—Snapshot, Transactional, or Merge. For this example, select Snapshot Replication, then click Next to continue.
3. If you need this to be an updateable snapshot, choose either Immediate Updating or Queued Updating on the Updateable Subscriptions screen (see Figure 12.7). If you choose both options, Immediate Updating will be used and Queued Updating will be set up as a fail-over option.



Figure 12.7: Choosing updateable subscription options.

4. On the Transform Published Data screen, you can choose whether data need to be transformed using DTS before they are distributed to your Subscribers. Click Yes or No, then click Next.
5. Next you need to specify the Subscriber types for this publication. The options on the Specify Subscriber Types screen are Servers Running SQL Server 2000, Servers Running SQL Server version 7.0, and Heterogeneous Data Sources.
6. Use the Specify Articles screen to browse the database and define which tables, stored procedures, and/or views you want to include in your publication. You can also define default Article Properties by selecting that option button. When you have finished defining your publication articles, click Next.

   Note The wizard might prompt you with an Article Issues screen if it finds any potential problems with article data types, such as IDENTITY columns that won't be transferred to the Subscriber. You should make a note of any of these problems.

7. Select a publication name and description when prompted. You might also check the box for "List this publication in the Active Directory" (see the section on this topic later in this chapter). Click Next to continue.
8. On the Customize the Properties of the Publication screen, select Yes if you want to define data filters or customize the remaining properties for the publication. Select No if you want to skip this step and do it later (if so, go to Step 12). Click Next to continue.
9. You can define the specific data rows and columns to be included with this publication. On the Filter Data screen, choose whether you want to filter data vertically (columns) or horizontally (rows), or select both. Click Next and a corresponding Filter Table Columns and/or Filter Table Rows screen will be displayed. Select the appropriate data, then click Next.
10. Select whether or not you want to allow anonymous subscriptions (see Figure 12.8 and the following discussion on setting up subscriptions) when prompted, and then click Next.

11. Select a job schedule for the initial synchronization process or set it to be run immediately after creating the publication.
12. Click Finish on the Completing the Create Publication Wizard screen. The publication will be created in a few moments.

Adding Subscribers

Use either the Push Subscription Wizard at the Publisher or the Pull Subscription Wizard at the Subscriber to create a Subscription. Both of these wizards are located in the Tools | Replication menu in Enterprise Manager. *In both cases, a publication must have been previously configured on the Publisher.*

Push vs. Pull Subscriptions

Push subscriptions can be created at the Publisher or Distributor using the Push Subscription Wizard. Push subscription metadata are stored on the site from which the push subscription was created. Push subscriptions require more Publisher overhead, so they should be used only in situations in which centralized management is a requirement and when the subscriber is always available to receive the pushed subscription. To create a push subscription for the Northwind snapshot publication created in the previous example:

1. From the Publications subfolder of the Replication folder in Enterprise Manager, right-click the publication name (for example, Northwind:Northwind), and select Push New Subscription to launch the Push Subscription Wizard. Click Show Advanced Options in this wizard, and click Next to continue.
2. In the Choose Subscribers screen, select your secondary database server, and click Next.
3. Enter a name for your subscription in the space provided in the Choose Destination Database screen. Use a descriptive name so that you can easily locate the subscription—for example, Server1_Northwind. Click Next.
4. In the Set Distribution Agent Location screen, select where you want to run the synchronization agent. You have the option to choose either the Distributor or the Subscriber. Click Next.
5. Choose the synchronization schedule for your subscription. This schedule should vary depending on how often the publication database is updated. Click Next.
6. In the Initialize Subscription screen, specify whether or not you want to initialize the snapshot for the description and, if so, where and when you want to do it. Click Next, and then Finish to create the push subscription.

Pull subscriptions (which might or might not be anonymous) can be created using the Pull Subscription Wizard at the Subscriber. Pull subscriptions should be used when subscribers might not be connected all the time or when there is a large number of subscribers and you want to reduce overhead at the Publisher or Distributor. To create a pull subscription to the Northwind publication, perform the following steps at your secondary server.

1. Right-click the Subscriptions subfolder in the Replication folder within Enterprise Manager, and select New Pull Subscription to launch the Pull Subscription Wizard. Check the Show Advanced Options in Wizard box, and click Next.
2. On the Look for Publications screen, select Look for Publications from Registered Servers, and then click Next.
3. In the Choose Publication screen, click the Northwind publication on your Publisher server, then click Next. (If the server is not listed, you need to register it in Enterprise Manager first. A button is provided on this screen for this purpose.)
4. You will be prompted to choose the database on the Subscriber where you want the subscription to put its data. Click the New button to create a new database. Name the new database **Northwind_Subscription** , and select appropriate size and location information. Click Next to continue.
5. From the Allow Anonymous subscription screen, choose No, This is a Named Subscription, and click Next.
6. On the Initialize Subscription screen, select Yes, Initialize the Schema and Data, and then click Next.
7. Specify how you want the snapshot to be delivered. Choose Use Snapshot Files from the Default Snapshot Folder for this Publication, and click Next.
8. Set up a synchronization schedule for the subscription, and then click Next and Finish to complete creation of the subscription.

Note that in order for anonymous subscriptions to be pulled, they must be either allowed or disallowed on a publication during its initial configuration at the Publisher (see Figure 12.8).

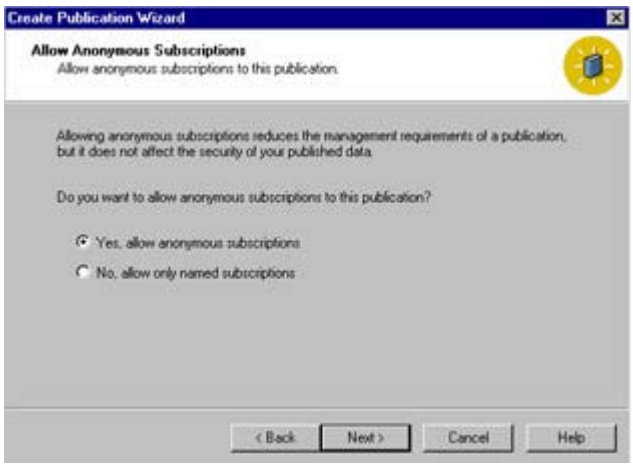


Figure 12.8: Setting the system to allow anonymous subscriptions.

Replicating Data Over the Internet

The Internet can be used as a transfer medium for all types of SQL Server 2000 replication. How you choose to replicate between your servers over the Internet depends on two main factors:

- **The intranet security measures you need to enforce** Most corporate networks have some sort of firewall or proxy solution in place to prevent unauthorized access. Virtual private networks (VPNs) are becoming commonplace; they allow users to connect from the Internet into a corporate network via a secure point-to-point "tunnel." Remote SQL Server 2000 Subscribers can use a VPN to connect to publications, or they can connect across the Internet via a secure port on Microsoft Proxy Server.
- **Your physical Internet connection speed and whether or not it is a permanent connection** Ideally, all your servers would be connected on the same network all the time, but if your connection is not permanent (in other words, it is dial-up), you obviously cannot use a static replication method. In this case, you can configure SQL Server 2000 Replication to use FTP to transfer snapshots back and forth.

Replicating Via a Virtual Private Network

The most secure and reliable way to connect your clients and servers across the Internet is with a VPN. A VPN is a secure point-to-point connection between a remote computer and a VPN gateway machine on your internal network. The gateway machine can be a server or other device (such as a Cisco router) running either Microsoft Point-to-Point Tunneling Protocol (PPTP) or Layer-Two Tunneling Protocol (L2TP). The remote computer, which must be running the same protocol, connects to the VPN with a secure, password-protected login (which is often the same as the Windows 2000 Domain login). Once connected, the remote computer receives the same rights and privileges afforded a local machine on the network. The external Internet connection and routing are transparent to the user.

VPN support is included with Windows 2000 and Windows 98/ME and is a free, downloadable add-on from Microsoft for earlier operating system versions. So, to set up replication from a remote SQL Server across a VPN, you need only configure that server's Internet connection and VPN connection, then run replication as you normally would with a local server. Keep in mind that if your VPN connection is not available when the servers try to replicate, replication will fail. It is common practice in this scenario to enable automatic reconnection in the event that the VPN connection is broken.

Replicating through Microsoft Proxy Server

Another possibility for networks running Microsoft Proxy Server for basic firewall protection is to set Proxy Server to allow access only via predefined secure ports. These TCP ports limit access to SQL Server's specific replication services only where permission is granted. The remote Subscribers can use pull subscriptions to request data from a Publisher behind the Proxy Server's firewall. Here are the basic steps for setting up SQL Server Replication under Microsoft Proxy Server:

1. Set up the Proxy Server and ensure that you have done the following:

   a. Disable IP Forwarding on the Proxy Server in the Windows Network Properties | TCP/IP Protocol | Routing tab.
   b. Install IIS on the Proxy Server and configure FTP Services at the standard port (21). Share the FTPRoot folder, and give full rights to the Publisher.
   c. Configure the Winsock Proxy Service to allow FTP site access only to specified users and disallow anonymous users.
   d. Create an inbound filter to TCP port 1433 on the Proxy Server, and assign access only to the Publisher/Distributor's MSSQLServer service account.
   e. Set the Subscriber to connect via FTP to the Proxy Server. The Subscriber's login account must be a member of the FTP Site Operators group local to the Proxy Server and must have permission to log in locally and at least read rights to the folder where the Snapshot will be stored.
2. Install the MS Proxy Client on the Publisher computer. Create a network connection from the Publisher/Distributor to the Proxy Server, using a secure NT Login account for the MSSQLServer service bound at port 1433. Set up an ODBC link to the Publisher on the Proxy Server.
3. Change the publication snapshot path to the UNC path of the FTP directory, and ensure that the Publisher has "Allow Snapshots to be downloaded using FTP" enabled in its properties.
4. Create a SQL Server connection (via TCP/IP port 1433) from the Subscriber to the Publisher/Distributor. Register the Publisher/Distributor server in Enterprise Manager on the Subscriber. Be sure to use a domain account rather than a SQL Server account to register the connection.
5. Create an anonymous pull subscription on the Subscriber and be sure to select "Yes, Use FTP to copy the Snapshot Files" under Snapshot Delivery. Set the FTP port to be the IP Address of the network card on the Proxy Server used to connect to the Publisher and port=21.

For more information on configuring Proxy Server for SQL Server Replication, see the white paper entitled *Configuring Proxy Server Replication* at http://msdn.microsoft.com/library/techart/proxyrep.htm.

Replication Via FTP

If you want to publish data on the Internet but cannot set up a VPN or a Proxy Server, there is a third option. You can use ODBC over TCP/IP for the SQL Server Publisher/Distributor and Subscriber server connections, then use a common FTP site for the actual snapshot file transfers. Internet publishing requires that you configure your application as follows:

1. Configure a Publisher or Distributor to listen on TCP/IP.
2. Configure a publication to allow Subscribers to retrieve snapshots using FTP.
3. Create a subscription to use FTP for retrieving snapshots.
4. Configure a subscription agent to use TCP/IP.

Configuring a Publisher or Distributor to Listen on TCP/IP

Publishers and Distributors must be enabled to listen on either TCP/IP (or the multiprotocol network library) before you can publish articles over the Internet. SQL Server 2000 uses TCP/IP Sockets to establish an ODBC connection across the Internet between a Publisher or Distributor and a Subscriber. Transactional subscriptions use the Distribution Agent to connect through the Internet to the Distributor; merge subscriptions use the Merge Agent to connect through the Internet to both the Publisher and the Distributor. FTP paths and ports can be specified as the snapshot folder location. Using publication properties, an FTP-configured server can be used as the snapshot folder location. You can also set the snapshot folder as the FTP home directory or set the FTP home directory as an FTP site.

Specify FTP Information (Enterprise Manager)

To set the snapshot folder as an FTP virtual directory:

1. From the Start menu, select Programs | Administrative Tools, and then select Internet Services Manager.
2. Right-click Default FTP Site (or whichever FTP Site you have configured), and select New | Virtual Directory.
3. Enter a Virtual Directory Alias (for example: Northwind_Snapshot). Click Next.
4. Type in or browse to the path to the snapshot directory (for example: C:\Microsoft SQL Server\Mssql\Repldata\Ftp), and then select Home Directory.
5. Check the Access Permissions for the FTP Directory option. Options are Read and Write. Click Next to complete the wizard.

To configure the SQL Server Client Network Protocol to support replication via FTP:

1. From the Start menu, select Microsoft SQL Server 2000, and then select Client Network Utility.
2. Double-check that TCP/IP or Multiprotocol appears in the Enabled protocols by order list, on the General tab.
3. If TCP/IP appears in the Disabled protocols list, select it, and then select Enable.
4. On the Aliases tab, in the Server alias field, type in the name of the server.
5. Select TCP/IP from the Network Libraries list, and in Computer name, overwrite the existing name with the IP address.
6. In Port number, overwrite the existing port number, if necessary. (It should be port 1433 unless you have customized your configuration to use a different port.)
7. Click OK to save your configuration.

Configuring a Publication to Allow Subscribers to Retrieve Snapshots Using FTP

Setting the @enabled_for_internet property to TRUE on the publication allows you to Internet-publish any publications you create. It also enables the Snapshot Agent to place the files associated with the initial snapshot into the FTP location specified in Publication Properties (see Figure 12.9). Using FTP, the Distribution Agent and/or the Merge Agent send snapshots of schema and data to the Sub-scriber via FTP. Snapshot images are recreated as exact duplicates of publications on the destination database. Once snapshot

files arrive at the Subscriber, the sending agent (Distribution/Merge) applies the files to the appropriate Sub-scriber tables. The agent then moves through each table performing the following:

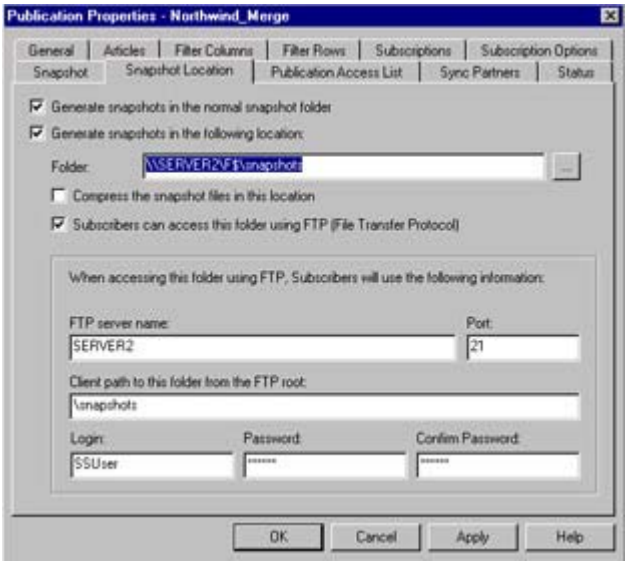


Figure 12.9: Setting up an FTP server as a snapshot location.

1. Taking out exclusive locks on a set of rows
2. Copying in the new rows
3. Releasing the locks on the rows
4. Repeating the process on the next blocks of rows

Other users should be able to continue using the tables, with little interruption, because the agent locks only a small number of rows at a time.

SQL Server Enterprise Manager allows you to configure a publication by selecting "Allow snapshots to be downloaded using FTP" on the Subscriptions Option tab (see Figure 12.9) of the *publication* Properties dialog box. The @enabled_for_internet property can also be set to TRUE when executing the following replication stored procedures:

- sp_addpublication
- sp_addmergepublication
- sp_changemergepublication
- sp_helpmergepublication

For example, to create an Internet-enabled snapshot or transactional publication, you might use the following syntax in Query Analyzer:

```
Exec sp_addpublication @enabled_for_internet=TRUE
```

Or, to add a merge publication with Internet subscribers:

```
Exec sp_addmergepublication @enabled_for_internet=TRUE
```

Configuring a Subscription to Use FTP to Retrieve a Snapshot

You must create a pull (or anonymous) subscription to the publication once the publication has been enabled for publishing on the Internet. All subscriptions are created the same way, whether or not they use the Internet. The only difference in subscribing to a publication over the Internet is that FTP addressing properties (FtpAddress, FtpPassword, FtpPort, and FtpUserName) must be configured for use by the Distribution Agent or Merge Agent.

FTP addressing can be configured through SQL Server Enterprise Manager on the Snapshot Location tab in Publication Properties (refer back to Figure 12.9).

Dealing with Replication Conflicts

Replication conflicts can occur during merge replication when two or more servers attempt to update the same database objects at the same time. For example, if users on more than one Subscriber change the same record in their local database copies, a conflict can arise when the changes are replicated back to the Publisher. The Merge Agent detects conflicts, once Publisher and Subscribers are reconnected and synchronization occurs, and then determines which data will be accepted and propagated to other sites. Data are accepted and propagated based on the resolver specified when the merge publication was implemented.

Conflicts can arise in merge replication when:

- Column-level tracking is in effect and changes are made to the same column(s) in the same row (using INSERT, UPDATE, or DELETE statements) in more than one copy.
- Row-level tracking is in effect and changes are made to a row in both replicas. (The columns affected in the corresponding rows need not be the same.)

Note A conflict typically occurs between updates made at different Subscribers and not necessarily updates made at a Subscriber and at the Publisher.

Replication Conflict Viewer

If a red “warning sign” icon appears in front of any publication or subscription names in Enterprise Manager, you might have a conflict requiring your attention. Click View Conflicts from a publication’s properties menu to launch the Replication Conflict Viewer (see Figure 12.10). Several tables are created by replication for the express purpose of reviewing information on conflicts and their corresponding resolutions. Conflicting rows are displayed by the Conflict Viewer, which allows you to see exactly what data caused a conflict so that you can set rules to avoid similar conflicts in the future and make sure that the correct conflict was resolved correctly.

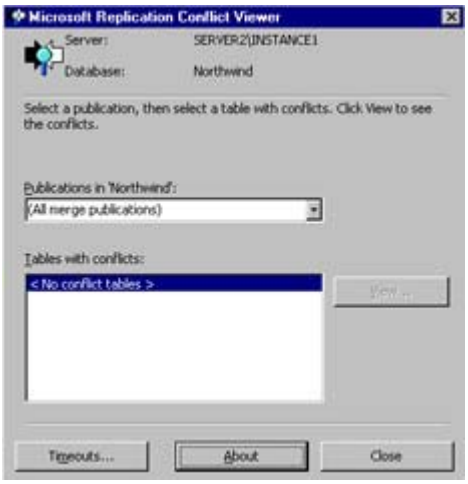


Figure 12.10: The Conflict Viewer.

Viewing Conflicts

The Conflict Viewer allows you to view the contents of any conflict tables that have been created during merge replication. A conflict table is created at publication setup for each table in a merge article. Conflict tables have the same structure as the tables on which they are based. Each row in a conflict table consists of a losing version of a conflict row (the winning version of the row now resides in the actual user table).

The sysmergearticles table contains a record for each replication article. Information stored for each article includes the corresponding name of the article on the Subscriber, the name of the article's local conflict table, the type of conflict tracking specified, and information about the custom resolver assigned to the article (if any).

A special log for rows that were deleted due to conflicts, called Msmerge_ delete_conflicts, is also generated during merge replication setup. This table contains conflict type and reason codes for deleted rows that were removed to achieve data convergence or that conflicted with an update and lost the conflict.

Row-Level vs. Column-Level Conflict Tracking

You can specify whether the Merge Agent tracks conflicts for an article in either of two ways: by row level or by column level.

If *row-level tracking* is selected, a change in any column in the row will result in the same row on multiple servers and result in a conflict. The conflict winner will overwrite the entire row of data on to the conflict loser. *Row-level tracking involves less tracking overhead.*

With *column-level tracking* , the same row can be updated on both servers, but conflicts will not occur unless the same column is updated. *Column-level tracking is usually more resource intensive.*

Use these steps to set row-level or column-level tracking for an article:

1. In the Create Publication Wizard, on the Specify Articles page, select the table that will be used as an article in your merge publication.
2. Click the Properties button for the selected table.
3. On the Properties tab of the article, on the General tab, under "When merging changes from different sources," select "Treat changes to the same row as a conflict for row-level tracking," or select "Treat changes to the same column as a conflict (changes to different columns in the same row will be merged)" for column-level tracking.

Resolving Conflicts

Merge conflicts can be resolved either by SQL Server's default conflict resolver or by a custom resolver. The default resolver is priority based; custom resolvers can be set up based on user-defined rules (implemented with stored procedures or COM+ applications) or using other specific Microsoft resolvers. Additionally, conflict resolution can be interactive, in which case conflicts are presented to the administrator at runtime in the Conflict Viewer for manual resolution. The Merge Agent launches the selected conflict resolver once a conflict is detected. Conflict winners are chosen according to user-specified resolvers for each article. Conflicts are resolved immediately after the resolver executes unless the interactive conflict resolver is used. The winning row is applied at the Publisher and Subscriber. The losing row is written to a conflict table named using the format conflict_ *PublicationName* _ *ArticleName* _ *usertablename.* For example, if you have a table named Customers that is published as an article named Northwind-Customers in the Northwind database publication, the corresponding conflict table would be named conflict_Northwind_Northwind-Customers_Customers.

In merge replication, conflict resolution takes place at the article level (property of an article) for a single row of data at a time. Publications composed of several articles can have different conflict resolvers serving different articles or the same conflict resolver serving one article, several articles, or all the articles that make up a publication.

When a subscription is created, it can be assigned a conflict priority value, or the priority value of the Publisher (see Figure 12.11) can be used. A subscription with a user-assigned priority value is called a *global subscription* . Global subscriptions are used when different Subscribers need different priorities. When you change a row in a global subscription, the subscription priority is stored in the metadata for the change. The changed row carries the priority value with it as it merges with changes at other Subscribers. This assures that a change made by a higher-priority subscription does not lose to a change made by a subscription with a lower priority.

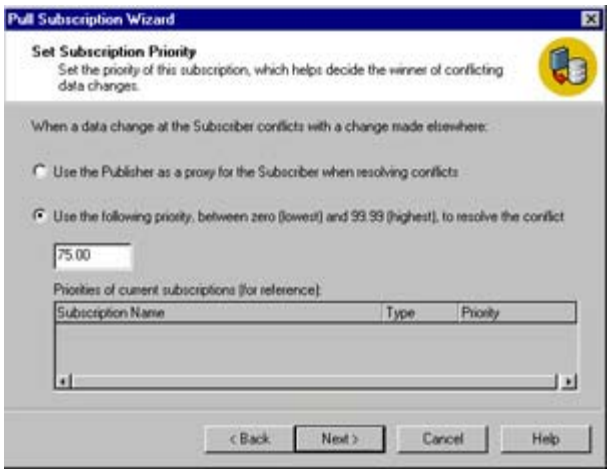


Figure 12.11: The Set Subscription Priority screen in the Pull Subscription Wizard.

A *local subscription* is a subscription that uses the priority value of the Publisher. This priority type is used when all Subscribers need to have the same priority, and the first Subscriber to merge with the Publisher wins the conflict. Use anonymous subscriptions when you expect a large number of Subscribers and you do not want to track them at the Publisher. No priority is assigned to a change in a row or a local subscription until the row merges with the other changes at a Publisher. Changes from the Subscriber are assigned the priority of the Publisher during the merge process at the Publisher and travel with that priority as they merge with changes at other Publishers and Subscribers.

Global subscriptions allow priority values to be preserved throughout the enterprise. They also provide a greater number of options and allow more detailed conflict resolution schemes than local subscriptions.

Local subscriptions, however, are also appropriate in a topology with several levels, where Subscribers are leaf nodes. Local Subscribers can be used only at leaf nodes, whereas any nodes that republish data must be global Subscribers.

Default Resolvers vs. Custom Resolvers

SQL Server has a default conflict resolver that takes effect for all published articles unless you specify a custom resolver in its place. Several custom resolvers are included with SQL Server, or you can write your own COM+ (Visual Basic or C++) application to handle merge conflict resolution. Each article can be configured to use a specific resolver, but once you define a resolver for a particular article, you must keep that definition consistent across all publications. When a push or pull subscription is created, you need to specify the way the default resolver will behave by selecting global or local for the subscription.

In SQL Server, subscriptions are defined as local by default, with a priority value of 0.00. The priority value selection corresponds to the option for using the priority value of the Publisher when a conflict occurs. Set the priority value in the Set Subscription Priority page in the Push Subscription Wizard or Pull Subscription Wizard. When the priority

values are the same for all Subscribers, the Publisher updates will win the conflict. Similarly, when a conflict exists between Subscribers that have the same priority value, the first Subscriber to synchronize will win the conflict.

Specific priority values can also be assigned to a subscription. Set the specific priority value to the Subscriber (from 0.00 to 99.99) in the Set Subscription Priority page in the Push Subscription Wizard or the Pull Subscription Wizard. When you select a priority greater than 0.0, you define a global subscription.

Global Subscriber priority values take precedence over priority values for any given local Subscriber, even if the values are the same.

If global and local Subscribers are connected to the Publisher, changes from global subscribers with priority values greater than 0.00 are synchronized with the Publisher first, and any conflicting changes from local Subscribers are rejected. If a local Subscriber connects to the Publisher before a global subscriber does, any subsequent conflicting changes from global Subscribers or other local Subscribers will be rejected. The Subscriber that is first synchronized with the Publisher assumes the priority value of the Publisher. The Publisher always wins the conflict; however, a custom resolver can override this rule.

SQL Server CE Edition Replication Features

SQL Server 2000 Windows CE Edition is a new addition to the SQL Server family. It allows native SQL Server database applications to be written for any device that runs Windows CE. These devices can subscribe to publications from your main SQL Server 2000 server via merge replication. Then, while these devices are disconnected, the data can be updated. When the devices are reconnected, changes are merge-replicated back to the main database. Good applications for these devices are warehouse inventory, package shipping and receiving, mobile health center data collection, or whenever the convenience of portable Windows CE devices can be implemented.

Figure 12.12 depicts replication with a SQL CE database. The SQL Server CE device uses Microsoft Internet Information Services as the go-between for the replication data transfer and for login validation. Security is standard Secure Sockets Layer (SSL) encryption. The SQL Server CE Agent (an ISAPI DLL) runs on the IIS server and communicates with both the SQL Server 2000 Publisher and the SQL Server CE Subscriber. The CE Client Agent sends the list of changes to the Server Agent, which stores them on the IIS server in a temporary file. Changes are reconciled with the Publisher using the conflict resolver you specified when you created the Publication. Once all conflicts are resolved, the Server Agent reads all the changes on the Publisher that have taken place since the client last synchronized and writes them to another temporary file on the IIS Server. Then the SQL Server CE Client Agent sequentially applies the data to the CE client database.

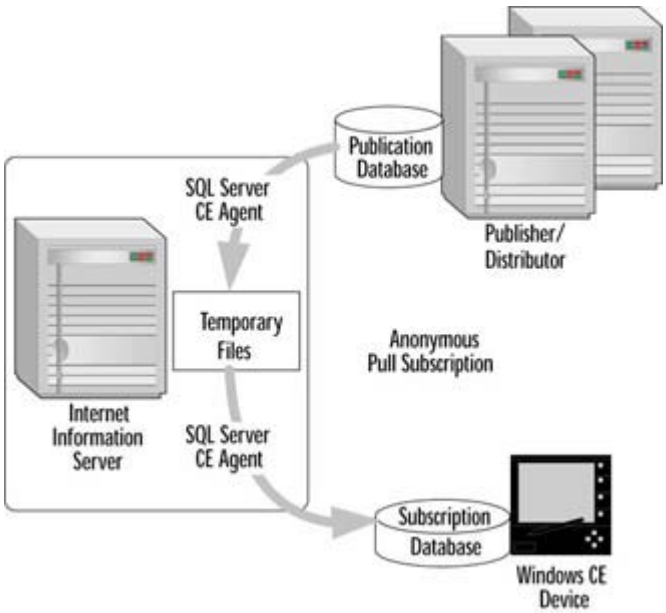


Figure 12.12: Windows CE replication.

Windows CE Subscribers

Windows CE Subscribers utilize *anonymous subscriptions* to merge publications. Subscriptions are administered at the Subscriber, and the Publisher stores no information regarding the CE Subscriber or the subscription.

Perform the following steps to set up replication for Windows CE clients:

1. From the Publisher, run the Create Publication Wizard and create the publication of the articles you want to share with your Windows CE Clients.
2. On the Specify Subscriber Types screen (see Figure 12.13), check the box next to "Devices running SQL Server CE." This will automatically enable anonymous subscriptions to this publication.



Figure 12.13: Setting up publication to Windows CE clients.

3. In your CE application, use the SQL Server CE Replication object to subscribe to the publication. When the application runs for the first time with the client connected to the Web server, the initial snapshot will be applied and the subscription database will be created on the client.

Note The following system tables will be created in the database on your Windows CE client the first time you set up replication: msmerge_replinfo, msmerge_tombstone, msmerge_genhistory, and sysmergearticles. These tables are used to track merge replication information and can be queried by your application. In addition, the following columns are added to each replicated table in the CE database:

s_Generation Keeps track of the rows that have changed since the last merge so that only changed rows are sent back to the Publisher.

s_RowLineage Keeps track of who made the change (any of the Subscribers or the Publisher).

Rowguid Used to uniquely track rows, as in SQL 2000. Added only if it doesn't already exist.

Replication and Active Directory Integration

New to SQL Server 2000 is the ability to browse and subscribe to publications using Windows 2000 Active Directory services. This functionality allows users to locate publications without necessarily knowing the name of the server or database instance of the Publisher.

Registering Publications in Active Directory

Before you can register publications in Active Directory, you need to make sure that you have Active Directory installed and configured correctly on the network. If this is the first publication you have set up, you need to add the SQL Server instance of the Publisher to Active Directory.

Browse to the SQL Server Instance object in Enterprise Manager. From the SQL Server Properties screen, click the Active Directory tab (see Figure 12.14), and then click the Add button. After a short pause, you should receive a message saying that the server was successfully registered with Active Directory.



Figure 12.14: Adding the server to Active Directory.

Note You must use a login belonging to the Local Administrators group on the server. If you do not have rights to add the server to Active Directory, a Connect to Servers dialog box will appear. If you do not successfully log in, you will receive an error.

Use the Refresh button to update the Active Directory attributes of the SQL Server instance. Use the Remove button to completely remove the SQL Server instance (and all its publications) from Active Directory.

To add an existing publication to Active Directory:

1. Ensure that the SQL Server service account has at least Power User privileges.
2. In Enterprise Manager, open the Replication folder, right-click the publication, and then select Properties.
3. In the General tab of the Publication Properties screen, check the check box in front of "List this publication in the Active Directory" (see Figure 12.15).

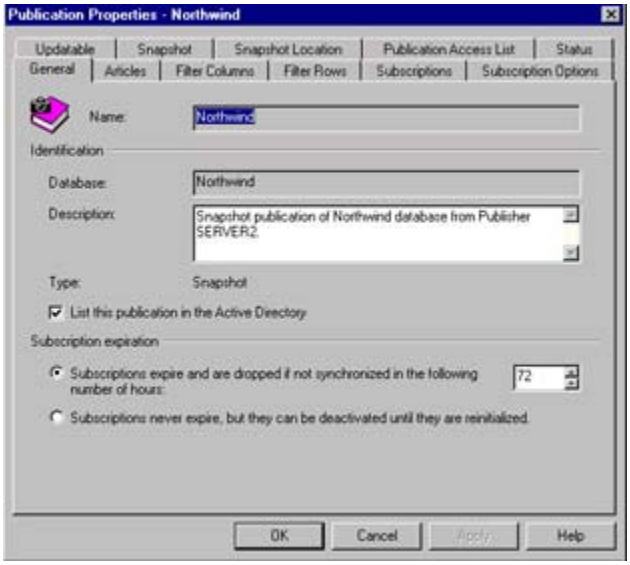


Figure 12.15: Adding an existing Publication to Active Directory.

Or to add a new publication to Active Directory:

1. Select "List this publication in Active Directory" from the Create Publication Wizard, Select Publication Name, and Description screen.

Tip You can also use the following system-stored procedures from T-SQL:

To add or remove the SQL Server Instance from Active Directory:

```
EXEC sp_ActiveDirectory_SCP @action='CREATE'
EXEC sp_ActiveDirectory_SCP @action='DELETE'
```

To add a new publication to Active Directory:

```
EXEC sp_addpublication @add_to_active_directory='TRUE'
Or
EXEC sp_addmergepublication @add_to_active_directory='TRUE'.
```

To add an existing publication to Active Directory:

```
EXEC sp_changepublication
@property=publish_to_active_directory, @value='TRUE'
Or
EXEC sp_changemergepublication @property=publish_to_ActiveDirectory,
@value='TRUE'.
```

Browsing and Subscribing to Publications in Active Directory

If a Subscriber has the correct permissions, he or she can browse the Active Directory to search for a publication to which to subscribe. In Enterprise Manager on the Subscriber, from the Pull Subscription Wizard:

1. On the Look for Publication screen, select "Look at publications in the Active Directory" or specify publication information.
2. From the Specify Publication screen (see Figure 12.16), click the Browse button. This will pull up the Find SQL Server Publications dialog box shown in Figure 12.17). Select or enter appropriate search criteria (if any), and click Find Now.

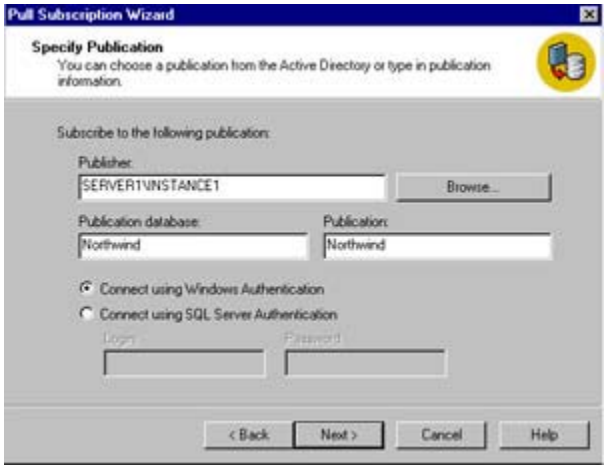
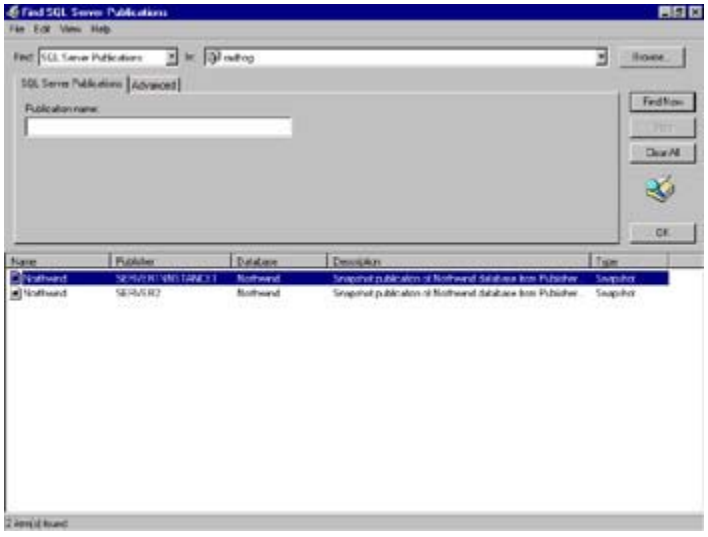


Figure 12.16: The Specify Publication screen in the Pull Subscription Wizard.


Figure 12.17: Browsing for a publication in Active Directory.

3. Select the publication to which you want to subscribe from the list of items returned, and click OK to continue setting up the pull subscription normally.

Note You can also use the Windows Synchronization Manager to browse and subscribe to publications using Active Directory. Windows Synchronization Manager is a tool that comes with Internet Explorer 5 and higher that is most commonly used for synchronizing e-mail and other applications when users dock a laptop. This utility provides a centralized management point for synchronizing with SQL Server Publishers as well.

Replication Performance Considerations

Replication is a very resource-intensive process on the server side, no matter what method you choose. Three of the ways you can improve performance are:

- Upgrade your server and network hardware.
- Modify your database design to minimize conflicts and speed queries and updates.
- Tune performance for each type of replication based on its specific requirements.

Hardware Upgrades

Performance of not only all types of replication but all other processes running on your SQL Server can be improved by upgrading your server's hardware. The following are some hardware recommendations that can improve replication performance:

- Add more memory to all of the replication servers. Use the min server memory option on the Memory tab of the Server Properties screen (or the sp_configure system-stored procedure) to set the minimum amount of memory available for SQL Server's use to avoid low memory during replication activities . Distributors require at least 16MB of memory. Use high quality disk subsystems. The speed of a disk subsystem is directly related to speed in which replication is completed. Additionally, if you require fault tolerance, you can mirror your hard drives (RAID1).
- Use multiprocessor computers. SQL Server 2000 allows replication agents to utilize additional processors on the server.
- Use a high-speed network. The speed of the network (100Mbps or faster) is directly related to the speed at which changes can be propagated to the Subscriber.

For Database Design Performance Tuning:

- Simply trigger logic or avoid using triggers on replicated tables.
- Avoid horizontal filtering. Instead, develop natural partitions of the table using DTS or custom data partitions with transformable subscriptions.
- Do not add additional indexes at the Subscriber. Throughput, with which changes can be applied to the Subscriber, can be significantly reduced by additional indexes.

Tip When you are aggregating most data at the Subscriber, it can be more efficient to create an indexed view at the Publisher. This can be published as a table to the Subscriber using transactional replication.

For General Replication Performance Tuning:

- Publish only required data.

- Utilize a separate disk drive for the snapshot folder and the transaction logs. Transaction writing time is significantly decreased when log files are stored on a different disk drive from the one on which the database exists.
- Update publications less frequently.
- Run the Snapshot Agent sparingly (only when necessary and at off-peak times).
- Use a single snapshot folder per publication. Snapshot files generated in more than one location require additional processing time.
- Consider compressing snapshot files. This practice can reduce storage requirements and save time transferring the files over a slow network or the Internet. However, compression can increase the time it takes to actually apply a snapshot.
- Use native bulk copy (BCP) mode to apply snapshot files to subscribers. This is most effective when ODBC or OLE DB Subscriber is not in use, when using transformable subscriptions, and when there is a large volume of data.
- Reduce the frequency of distribution when replicating to numerous Subscribers.
- Use pull and anonymous subscriptions when there are a large number of Subscribers to avoid storing subscription information at the Publisher.
- Run agents continuously instead of stopping and starting them on very frequent schedules.

How to Enhance Snapshot and Transactional Replication Performance

Use the following performance tips to enhance both transactional and snapshot replication:

- Configure the Distributor on a dedicated server to reduce overhead on the Publisher.
- Subscribing to all articles in a publication allows the Distribution Agent to use an optimal query during synchronization.
- Use stored procedure replication for large numbers of affected rows. This method is significantly faster than sending the update/delete as individual row changes.
- Minimize the retention period for transactions and history to save disk space.
- Don’t unnecessarily reinitialize subscriptions, and don’t let subscriptions expire that you plan to continue using but use infrequently (increase expiration dates to prevent this).

How to Enhance Transactional Replication Performance

Transactional replication performance can be enhanced if you:

- Increase the Log Reader Agent read batch size (default is 500 transactions at a time).
- Minimize both the log history and the retention period to save disk space.
- Use customized stored procedures for inserts, updates, and deletes at Subscribers.

How to Enhance Merge Replication Performance

Some ways to enhance merge replication performance are:

- Modify your database design. Limit the use of text and image columns. Use indexes on columns that are used in article and join filters.
- Create a ROWGUIDCOL column for each published table prior to generating the initial snapshot.
- Increase Merge Agent batch sizes.
- Limit complexity of article filters, and don't overuse join filters.
- Use column-level merge tracking if bandwidth is limited.
- Use global subscriptions for conflict resolution prioritization.
- Reindex merge replication system tables on occasion.
- Limit/control simultaneous agent processing.

Replication Backup Strategies

Replication technology requires that you incorporate additional backup and restore strategies in addition to your normal SQL Server 2000 backup strategy. Failures can occur in different places and with different effects, depending on the replication method you use. To be able to successfully restore your replication setup, you need to back up some or all of the following components on a regular basis:

- **Publisher** The most important replication component for which to keep regular backups is the Publisher. If possible, perform a full database backup of the publication database(s), and preferably the Master and msdb system databases as well, at least once daily, and perform incremental backups and transaction log backups more frequently. Frequency of backups depends on the frequency of data change in your unique environment, of course.
- **Distributor** In order to properly back up the Distributor, you must back up the distribution database, the msdb database, and the Master system database. Using this procedure allows you to recover from most failure types without having to recreate publications or reconfigure replication. Backing up the Distributor allows you to recover faster in the event of a Publisher or Distributor failure, because replication does not need to be reestablished. Where transactional replication is concerned, SQL Server 2000 automatically handles the coordination between backing up the publication database and the distribution database. Once the distribution database is backed up, make transaction log backups and differential database backups.
- **Subscriber(s)** In most cases, replicated data must be restored at the Publisher or Distributor, so there is no real need to back up those data on the subscriber. However, regular backups of each subscription database and relevant system databases at the Subscriber could circumvent the need to reinitialize subscriptions in the event that recovery is required. If the Subscriber uses pull subscriptions and if there is a need for restoration following a total system loss, back up the msdb and Master system databases. Once the subscriptions database has been backed up, make transaction log backups and incremental database backups.

When planning your backup strategy, keep in mind that the more complex your strategy, the more difficult and time-consuming it will be to restore. Backup of all databases is required only if you need to be able to restore replication data from the backup to minimize downtime and the likelihood of data loss.

SQL Server replication has the ability to reinitialize one or more subscriptions on demand, which—together with regular Publisher database backups—provides a simple recovery strategy. However, you can limit regular backups of your publication databases because SQL Server replication scripting provides a method of reestablishing replication when the entire replication environment needs restoration.

If the Publisher and Distributor are synchronized, another strategy is to back them up but not the Subscribers. This plan allows you to completely restore a replication environment.

Subscribers need to be reinitialized after the restore, which is time-consuming but not fatal. Backing up a Subscriber can reduce the time it takes to recover from a failure of the Subscriber, but it is entirely optional.

Any time replication objects are added or changed, it is a good idea to back up both the publication and the distribution databases simultaneously. Be sure also to restore the same versions of both databases.

Regardless of the backup strategy you select, you should always maintain a current script of your replication settings, in addition to any regular backups, in a safe location. Should a total server failure occur, you can then use the script to aid in recovering replication to a new server by modifying the script to change the server name references.

Backing Up and Restoring Snapshot Replication

Since snapshot replication generates and delivers a complete snapshot of propagated changes for the publication, you can back up the publication database less frequently than is necessary with transactional replication or merge replication. A good rule of thumb is to back up the publication database when changes are made to the existing publication properties or when new publications are added.

Backup and Restore of Transactional Replication

When the publication database and the distribution database are restored to a consistent point in time, recovery of transactional replication from a loss of either can be accomplished without having to reinitialize subscriptions or reconfigure replication. SQL Server 2000 automatically handles the coordination of the backups of the two databases, whereas previous versions required manual, simultaneous back up of both databases while ensuring that no changes were made during backup.

Setting the "sync with backup" option at the Publisher and the Distributor directs the Log Reader Agent to not propagate any transactions to the distribution database without first backing them up at the Publisher. This ensures that the Transaction Logs will always be kept synchronized during a database backup and that they will be able to be restored to a consistent point in time.

Warning Publishers running on versions prior to SQL Server 2000 cannot use the "sync with backup" option. Even if this option is set, it will have no effect on the distribution database.

Using Log Shipping as a Backup to Transactional Replication

Log shipping is a new feature that allows the administrator to schedule automated movement of transaction logs to another server, which can then function as a warm "standby" server. (See Chapter 3 for details on enabling this feature.) SQL Server 2000 Enterprise Edition must be running in order to use log shipping. You can configure

transactional replication to assist in the log shipping process. There are two modes for replication and log shipping working in concert:

- **Synchronous mode** When the "sync with backup" option is set on the publication database, the Log Reader Agent synchronizes with the publication database backup. In this mode, the Log Reader Agent propagates transactions from the Publisher to the distribution database only if they have not been backed up. This choice is advantageous because all replication servers can remain synchronized after failing over to the log-shipping standby Publisher.
- **Semi-Synchronous mode** Use this mode if the increased latency of synchronous mode is unacceptable. If the warm standby Publisher and the Subscribers are not be synchronized because the performance of the Log Reader Agent and the backups are not synchronized, that would be another reason to choose this mode. The Semi-Synchronous mode allows transactions to be propagated to the Distributor and then to Subscribers, regardless of whether or not they have been backed up on the Publisher and shipped to the warm standby. Replication can be restarted even though the Publisher and the Subscribers are now out of synchronization.

Backing Up and Restoring Merge Replication

Since merge replication effectively creates backups of data to multiple publishers or subscribers, restoration of a Publisher or Subscriber can be achieved by resynchronizing with another server that has a copy of the same articles or one that has had the articles log-shipped to it. This can be done even if an up-to-date backup is unavailable.

There is no general requirement to restore publication and distribution databases to the same point in time, because merge replication stores change tracking metadata directly in your publication and subscription databases.

How to Restore a Replicated Database from Backup

In general, you cannot keep replication settings if you restore to a different server/database than your backup. Replication metadata are automatically removed by the Publisher or Merge Subscriber server when you perform a full restore to a different database or server. Therefore, you will need to completely recreate your replication schema (publications and subscriptions to them) in order to restore to a new server. Keep this in mind if you plan to migrate to a new server in the future.

When to Perform Additional Backups

Any time you make modifications to your replication setup, it is a good idea to make a backup in addition to your normal scheduled backup. It is also a good idea to keep a "before" backup copy around in case you accidentally make a change that has adverse effects. In general, you should always make a backup of all related databases when you

are enabling or disabling replication. The following is a list of some of the other actions after which you should make additional backups.

Actions Requiring Additional Backups at the Publisher:

Back up the publication database after you:

- Create new publications or drop publications.
- Change any publication's properties.
- Add articles to or drop articles from an existing publication.
- Change any article's properties, including the conflict resolver it uses.
- Enable or disable replication.
- Reinitialize all subscriptions for a publication.
- Change the schema or structure of any published table.
- Replicate using a manual script.
- Clean up Merge replication metadata (by running sp_mergecleanupmetadata).

Back up the Publisher's msdb database after you:

- Enable or disable publishing of any database.

Back up the Publisher's Master database after you:

- Enable or disable publication of a database.
- Create the first or drop the last publication for any database.
- Enable or disable a Publisher to use a particular Distributor.

Actions Requiring Additional Backups at the Distributor:

Back up the distribution database after you:

- Create or modify Replication Agent profiles or Modify Agent profile parameters.
- Change the Replication Agent properties (including schedules) for any push subscriptions.

Back up the Distributor's msdb database after you:

- Add or drop a distribution database.
- Create or modify replication agent profiles.
- Modify any replication agent profile parameters.
- Change replication agent properties (including schedules) for any push subscriptions.

Back up the Distributor's Master database after you:

- Add or drop a distribution database.

- Enable or disable a Publisher to use that Distributor.
- Enable or disable a Subscriber to use that Distributor.

Actions Requiring Additional Backups at the Subscriber:

Back up the subscription database after you:

- Change any subscription's properties.
- Change a subscription's priority at the Publisher.
- Drop a subscription.

Back up the Subscriber's msdb database after you:

- Change replication agent properties (including schedules) for any pull subscriptions.

Back up the Subscriber's Master database after you:

- Create the first or drop the last publication for any database.
- Enable or disable a Subscriber to use a particular Distributor.

When you do a complete database backup in SQL Server 2000, all user and system tables from the Master database, including sysobjects, are backed up as well. Additionally, the transaction log files, including everything after the last log read transaction, are backed up. When you do a restore to the same server or database from which you did the backup, SQL Server 2000 restores all database and log files to the exact moment in time the backup took place.

After restoring a publication database, SQL Server reads the category column from the master.sysdatabases table to determine if it should preserve any replication settings. So, most of the time your publication database settings will be maintained when restoring it to the same server and database from which it was backed up.

Warning If you have to completely recreate a publication database, your replication settings will be lost during the restore unless you run the following stored procedure: EXEC sp_replicationdboption.

In addition to backing up the publication database, you must back up the corresponding database on the Distributor for that Publisher. Both databases could be needed to restore replication while maintaining transactional integrity. In addition, you need to back up the replication working folder and temporary snapshot folders on the Publisher or Distributor to reduce resynchronization time to Subscribers after a restore.

Distribution databases are database and server name dependent, so they cannot be restored to a location other than the one from which they were backed up. You should never restore a distribution database without restoring the publication database also.

Merge Subscribers require the same considerations for backup and restoration as Publishers because they have many common settings that need to be preserved.

When you back up a Subscriber using transactional replication, the MSreplication_subscriptions table is also backed up. This table keeps track of the transaction that was last received at the Subscriber and maintains transactional integrity for the restoration. Also after restoration, run the sp_vupgrade_subscription_tables stored procedure to check that all objects necessary for replication have been restored properly and are the correct versions.

Summary

Replication Publishers are SQL Server databases that share publications consisting of specific database articles (tables, stored procedures, or views). Distributors monitor the replication process and store interim data and metadata as well as history and error logs. Subscribers are remote machines that can either pull a subscription to a publication from the Publisher or have a subscription pushed to them from the Publisher. All the data movement is controlled by processes run by individual agents, all of which are controlled and hosted by the SQL Server Agent.

There are three main methods of replication in SQL Server 2000. Snapshot replication copies the entire publication from a Publisher to a Subscriber each time it runs. Transactional replication copies an initial snapshot, then only updates are copied subsequently. Merge replication allows both Publisher and Subscriber to merge updates back and forth to one another. Merge replication includes fully customizable conflict resolution to handle conflicting database changes coming from Publisher or any of the Subscribers.

Plan your replication schema according to your company's needs in terms of user and data physical location, who modifies data and how often, and network performance. Care should be taken when designing a database that will be replicated. Avoid using IDENTITY and TIMESTAMP columns for primary keys, since they are not unique across servers; use the UNIQUEIDENTIFIER, which is a globally unique identifier (GUID).

Create a good plan of action for backing up and restoring the Publisher, and keep Distributor and Subscriber databases as well as the transaction logs and replication control data (MSDB and Master databases) on each server. Log shipping can be used as a companion to other types of replication backups to keep a warm standby server available.

Replication performance can be enhanced in a variety of ways, including physical hardware upgrades (memory, processors, and hard disks), replication-specific configuration changes, and database object design considerations.

SQL Server Replication can grow with your company to meet whatever challenges lie ahead. It includes support for replicating with Windows CE devices running SQL Server 2000 CE Edition, compatibility with older versions of SQL Server, and integration with a variety of non-SQL databases, including Microsoft Access, Oracle, and DB2.

Replication in SQL Server 2000 is now more powerful than ever. Enhancements include queued updating, which allows a Subscriber to work while disconnected from the Publisher, with changes being queued up until the connection is restored. Subscribers can now request locally customized versions of publications using transformable subscriptions. Replication also supports new SQL Server 2000 features such as indexed views, user-defined functions, new data types, and multiple SQL Server instances.

Finally, SQL Sever 2000 replication publications can be made available for browsing and subscribing by clients using Active Directory integration.

FAQs

Q. We have a large data warehousing application that is used by the sales department for analyzing market trends on a monthly basis. All the sales staff analysis is done using a read-only, third-party reporting tool. We need to move data to the warehouse from another server containing the current sales and order-entry system without impacting performance. What type of replication could we use, and why?

A. Use snapshot replication. Because the sales data do not have to be sent to the data warehouse immediately, since the sales analysts require data only as recent as the previous month, you could set up snapshot replication to copy the order-entry data infrequently (once a week or so) at off-peak times.

Q. One of our warehouses is in a remote location with dial-up Internet access (56K modem) only, and we need to keep its inventory synchronized with our corporate office. To take inventory, we use Windows CE handheld barcode terminals to walk around the warehouse and scan labels on merchandise. What type of replication could we use, and why?

A. Use merge replication to Windows CE. Merge replication can be configured to interface with Windows CE clients over a Web server. Clients do not have to be connected all the time and can dial in to upload or download data as required.

Q. Our video store has three branches, but we allow customers to rent movies from any of the stores using their membership ID cards. Since we limit customers to three rentals at a time, we need to check customers' current rental quantity across all the stores before renting them any more movies. We have a fairly reliable 128K ISDN connection between sites, but they may occasionally be disconnected from one another. However, we still need to be able to access the customer rental records at each site. What type of replication should we use, and why?

A. Use transactional replication with immediate updating and queued updating set as a fail-over. Transactional replication sends updates across the wire only after initial snapshot generation, so it can be used effectively over the relatively slow ISDN link. Since all three stores will be updating in real time most of the time, updating can be set to immediate. Queued updating should be set in the event that the ISDN line is down, so rental transactions can be queued up and batched in later.

Q. I want all my product part numbers to be system-assigned IDENTITY 4-digit numbers, but I need to assign part numbers from two SQL Servers on our network and merge-replicate the data between them. I could have up to 2000 part numbers per site, and I might add another Subscriber later. How can I manually enforce different IDENTITY column ranges at the Publisher and Subscriber during merge replication so that I don't get conflicts?

A. You'll need to create or alter the PART tables at the Publisher and Subscriber to start at different base part numbers to ensure that ranges don't overlap. Also, you'll need to use the NOT FOR REPLICATION property to tell the servers not to reassign IDENTITY values during replication.

Use code similar to the following to create the PARTS table on each server:

At the Publisher:

```
CREATE TABLE parts (PARTNO INT IDENTITY (1000, 1) NOT FOR
REPLICATION PRIMARY KEY)
```

At Subscriber #1:

```
CREATE TABLE parts (PARTNO INT IDENTITY (3001, 1) NOT FOR
REPLICATION PRIMARY KEY)
```

At Subscriber #2 (future):

```
CREATE TABLE parts (PARTNO INT IDENTITY (5001, 1) NOT FOR
REPLICATION PRIMARY KEY)
```

Chapter 13: Programming Tools and Technologies in SQL Server

Introduction

Every release of SQL Server results in a more robust programming environment. SQL Server 2000 continues with that tradition, including many enhancements that strengthen the productivity and programmability of SQL Server solutions. Additional data types, such as the table data type, increase performance and programmability over traditional methods. User-defined functions allow developers to extend the programming environment with custom functions for reusability and simplified application logic. Referential integrity and trigger enhancements will help SQL developers manage data change events in SQL Server.

Productivity enhancements in SQL Server 2000 include the new Meta Data Services and an enhanced Query Analyzer tool. Meta Data Services in SQL Server 2000 offers shared application and modeling information with support for standard Component Object Model (COM) and Open Information Model (OIM) interfaces. One of the most evident visual changes is in the SQL Query Analyzer tool. Query Analyzer now includes an embedded object browser and dozens of T-SQL templates for creating everything from tables to adding linked server logins. These features will assist both new and seasoned T-SQL developers in taking advantage of SQL Server 2000's latest features. The SQL debugger utility addition to SQL Server 2000 offers stronger integration and reliability—a long overdue tool for T-SQL programmers.

This chapter reviews the programming tools and technology enhancements in SQL Server 2000. The chapter includes examples using each of these new features, allowing you to begin taking advantage of these enhancements in your applications. A review of data access technologies such as ADO, ODBC, and OLE DB will allow you to efficiently integrate your client applications with SQL Server. The end of this chapter provides you with an overview of the programming models available to extend Analysis Services, Data Transformation Services, Replication, and Meta Data Services in SQL Server.

Overview of SQL Server Programming

If you are a developer in the information technology field, you work with databases. Data access technologies have been evolving over the past 30 years, since the inception of the relational database model in the early 1970s. As recently as 10 years ago, data access meant reading a green screen terminal. Microsoft introduced Open Database Connectivity (ODBC) in 1992, providing the architecture for a common programming interface to database systems from multiple vendors. ODBC currently supports over 170 database access drivers.

This standard worked fine if your data source was a traditional data-base, but it could not be easily applied to nontraditional data sources such as your company's e-mail system. Microsoft's Object Linking and Embedding Database (OLE DB) technology filled this gap by offering a high-performance, data access interface to all types of data storage systems. ActiveX Data Objects (ADO), the latest data access technology, provides an efficient and intuitive interface for developers connecting to an OLE DB or ODBC data source; it can manipulate data from any Windows-based development platform, such as Microsoft Visual Basic or Active Server Pages.

No matter how well your application's user interface is designed, it will not perform well if the database is not responsive; the success of your application depends on its performance. Your database is the foundation on which the rest of the project is based. Proper database development involves proper planning. A developer should be familiar with database optimization techniques. A well-placed index can vastly improve performance, whereas a poorly written query can drag it down. Therefore, a working knowledge of Transact-SQL (T-SQL) is also essential. T-SQL is a dialect of American National Standards Institute (ANSI) SQL, which is a scripting language used to retrieve and manipulate data.

Database programming discussions are normally concerned exclusively with database access for application development. This exclusivity implies that developers are disconnected from database administration and management. That might be true for entry-level developers, but as your career builds, so do your responsibilities. You can administer and maintain your database using the Enterprise Manager and various wizards. If your duties involve performing rudimentary tasks on a single database, that might be sufficient. However, large organizations, in which multiple databases need tending and resources are limited, bring additional demands. Any task that you can perform through Enterprise Manager, you can also do using T-SQL. You can script common tasks such as backing up a database or performing a bulk insert. For more complex database or server management tasks, software development tools such as Microsoft Visual Basic or Visual C++ can be used to implement the features of SQL-DMO, a suite of objects designed to programmatically perform tasks such as adding users or getting a list of databases from a server. Other administrative tasks, such as replication and data transformation services, can be automated in a similar manner.

This chapter is not intended solely for developers. Discussions about database administration are normally concerned with management, administration, and optimization. Often, it is assumed that database administrators (DBAs) have no need of programming skills. Experience proves otherwise. With some programming skills, DBAs can ease their daily workload by taking advantage of SQL Server's integrated framework of administrative objects and services.

New Programming Features in SQL Server 2000

In this section, we explore the new programming enhancements in SQL Server 2000—from new data types to trigger enhancements. Developers will enjoy some of the enhancements to Query Analyzer, including a data-base object browser and a long-awaited debugger. User-defined functions provide an interesting and useful alternative to stored procedures. Referential integrity is now much easier to maintain with cascading updates and deletes. Views in SQL Server 2000 can be treated more like tables. The new INSTEAD OF trigger allows you to update and insert to views built from multiple tables. Indexes can now be placed directly on views to optimize performance. Using the new Meta Data Services feature, reusable scripts and components can be shared across a company.

Data Types

Three new data types were introduced with SQL Server 2000. The *bigint* data type extends the limitations of the original *int* data type, having a value range of -231 (-2,147,483,648) through 231-1 (2,147,483,647). The *sql_variant* type allows for a wider range of storage options in the same column. The *table* data type temporarily stores rowsets. It can be used as the return value of a user-defined function; however, it cannot be used to define a column.

Bigint

The *bigint* data type is a 64-bit integer ranging from -263 (9,223,372,036,854,775,808) to 263-1 (9,223,372,036,854,775,807). As expected, *bigint* can be used anywhere the *int* data type is used. This is especially handy for increasing the range of possible values of a column with the IDENTITY attribute as follows:

```
CREATE TABLE TstBigInt
(TestID bigint IDENTITY,
TestValue bigint)
GO
```

Tip Within SQL Server 7.0, the @@IDENTITY function retrieves the identity value of the last row inserted into a table. However, if the table inserted has a trigger on it that inserts into another table, this value is misleading. There are two new functions to circumvent this problem. SCOPE_IDENTITY returns the last inserted identity value within the current scope. Since triggers do not occur in the same scope as the current connection, the identity value is accurate. The IDENTITY_CURRENT *(tablename)* function can be used to return the last identity inserted into a specific table.

Two new functions were added in support of this data type. BIG_COUNT works just like the COUNT aggregate function, except it returns a *bigint* . The @@ROWCOUNT function returns the number of rows affected by the last executed SELECT, INSERT, UPDATE, or DELETE statement. The new @@ROWCOUNT_BIG function serves the

same purpose but returns a *bigint* . If the value returned by one of these functions is within the normal range of the data type storing the value, SQL Server will automatically convert the data type:

```
DECLARE @iCount int

SELECT @iCount=BIG_COUNT(*) from TestBigInt

PRINT @iCount
```

Existing standard functions return *bigint* data types if the return value is outside the range of an *int:*

```
DECLARE @biMax bigint


INSERT INTO TestBigInt VALUES (9213138212)
SELECT @biMax=MAX(TestValue) FROM TestBigInt


PRINT @biMax
```

Sql_variant

The *sql_variant* data type is, as you'd expect, more complex. It is similar to the Visual Basic variant data type. It cannot store certain data types, which include *text, ntext, image, timestamp,* and *sql_variant* types. You could place a unique index on a *sql_variant* column or use it as a primary key with limitations. If a *sql_variant* is used as a primary key, it cannot exceed 900 bytes in length. In addition, it does not support the IDENTITY attribute.

Metadata is stored along with the value, including the base data type, maximum size, scale, precision, and collation. These properties are accessed with the new SQL_VARIANT_PROPERTY function, as follows:

```
DECLARE @vTestVariant sql_variant

SET @vTestVariant = CAST(9213138212 as bigint)

SELECT SQL_VARIANT_PROPERTY(@vTestVariant, 'BaseType') as BaseType,
    SQL_VARIANT_PROPERTY(@vTestVariant, 'MaxLength') as MaxSize,
    SQL_VARIANT_PROPERTY(@vTestVariant, 'Scale') as Scale,
    SQL_VARIANT_PROPERTY(@vTestVariant, 'Precision') as Prec,
    SQL_VARIANT_PROPERTY(@vTestVariant, 'Collation') as Coll,
    SQL_VARIANT_PROPERTY(@vTestVariant, 'TotalBytes') as Tbytes
```

If the value of a *sql_variant* variable is null, the base type is also null, even if the variable is explicitly cast. The following code will return NULL even though @iInt is declared as an *int* . If @iInt is assigned a literal value, the code will return with the base data type. This functionality is by design.

```
DECLARE @vVarTest  sql_variant
DECLARE @iInt    int

SET @iInt = NULL
SET @vVarTest = @iInt

SELECT
SQL_VARIANT_PROPERTY(@vVarTest, 'BaseType')
```

Comparing *sql_variant* values takes some planning and consideration. Data types are grouped by data type family, then by data type hierarchy, as outlined in Table 13.1 .

Table 13.1: Data Type Family Groupings

Data Type Hierarchy	**Data Type Family**
sql_variant	*sql_variant*
datetime	*datetime*
smalldatetime	*datetime*
float	Approximate number
real	Approximate number
decimal	Exact number
money	Exact number
smallmoney	Exact number
bigint	Exact number
int	Exact number
smallint	Exact number
tinyint	Exact number
bit	Exact number
nvarchar	Unicode
nchar	Unicode
varchar	Non-Unicode
char	Non-Unicode
varbinary	Binary
binary	Binary
uniqueidentifier	*uniqueidentifier*

When two *sql_variants* within the same family are compared, SQL Server 2000 promotes the data type lower in the hierarchy to the data type farther up. For example, if an *sql_variant* with an *int* base type is compared to a *sql_variant* with a *bigint* base type, the *int* will be promoted to a *bigint* . However, when two *sql_variants* in different data type families are compared, they must be explicitly cast to the same family for a proper

comparison. Unicode data is in a separate family from exact number data. If two *sql_variants* in these families are compared, the value in the exact number family will be evaluated as greater than the other, as illustrated in the following example:

```
DECLARE @vTestVariantA sql_variant
DECLARE @vTestVariantB sql_variant

SET @vTestVariantA = CAST('100' AS char(3))
SET @vTestVariantB = CAST(50 AS int)

IF @vTestVariantA > @vTestVariantB
   PRINT 'Without CAST: 100 > 50'
ELSE
   PRINT 'Without CAST: 100 < 50'

IF CAST(@vTestVariantA as int) > CAST(@vTestVariantB AS int)
   PRINT 'With CAST: 100 > 50'
ELSE
   PRINT 'With CAST: 100 < 50'
```

This code generates the following output:

```
Without CAST: 100 < 50
With CAST: 100 > 50
```

In the first comparison, the values are cast as data types in two different families. The @vTestVariantA is in a data type family below @vTestVariantB. In the second comparison, the variables are cast to the same data types for a proper comparison.

Table

The *table* data type is unique in that it cannot be assigned as the data type of a column. It is primarily used to return rowsets of data from stored procedures or functions. It has a distinct advantage over temporary tables because it is created in memory instead of in the *tempdb* database. This makes it ideal for returning small to moderate rowsets. You should be careful not to return enough rowsets to fill a significant amount of memory on the database server. Declaring a *table* data type is similar to using the CREATE TABLE statement:

```
DECLARE @tEmpShort table
(EmployeeID int PRIMARY KEY,
 FirstName nvarchar(10),
 LastName nvarchar(20))

INSERT INTO @tEmpShort
SELECT EmployeeID, FirstName, LastName
FROM Employees

SELECT * FROM @tEmpShort
```

The returned rowset can be modified like a normal table. You can insert, edit, and delete rows in the table, but since it is created in memory, the underlying table is unaffected. The *sql_variant* and *bigint* types can be used as data types in a table variable, but user-defined data types cannot be used.

Query Analyzer

Developers will find a number of notable and immediately useful enhancements in Query Analyzer. The Object Browser and templates (see Figure 13.1) are significant productivity enhancements. We also cover Query Analyzer's built-in SQL debugger.

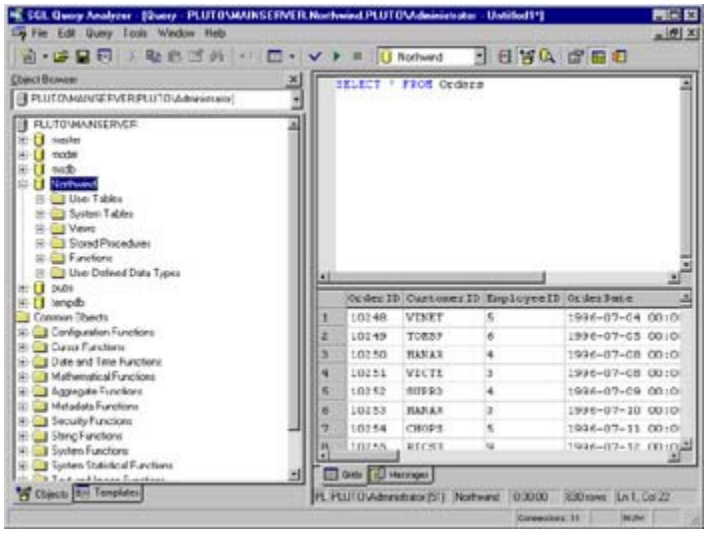


Figure 13.1: Query Analyzer Object Browser and templates.

When writing a query with SQL Server 7.0, you had to switch between a display of the database schema in Enterprise Manager or use the *sp_help* stored procedure to look up the name of a column. This is no longer necessary. The Query Analyzer Object Browser allows you to drill down into each individual object, from tables to indexes to stored procedures. Individual column types, as well as various other table-related objects such as triggers and indexes, are displayed beneath each table. User-defined functions are available under the Functions folder.

In SQL Server 7.0, you could automatically generate scripts for CREATE and DROP statements for objects through Enterprise Manager. Now that same functionality, and more, is available through the Object Browser. You can speed development by having Query Analyzer automatically create scripts for you. Let's say that you want to create a basic insert statement for a single table. Have Query Analyzer automatically generate the template for inserting into the Region table as follows:

1. Open Query Analyzer and connect to your local server.
2. If the Object Browser is not already visible, open it, either by clicking on the Tools menu and selecting Object Browser or by pressing F8 .
3. Expand the Northwind database node.

4. Expand the User Tables node.
5. Right-click the dbo.Region.
6. Select Script Object to New Window As ….
7. Select Insert.

A new window containing the following template code will be generated:

```
INSERT INTO [Northwind].[dbo].[Region]([RegionID], [RegionDescription])
VALUES(<int>, <nchar(50)>)
```

Of course, it is up to you to fill in the values to insert.

You can also easily create a scripted call to a stored procedure, which will contain the variable declarations for the parameters and proper assignments. We will create a scripted call to the CustOrderHist stored procedure. The following steps assume that the Norwind database is already selected in the Object Browser:

1. Expand the Stored Procedures node.
2. Right-click the CustOrderHist stored procedure.
3. Select Open.
4. You will be presented with the dialog box shown in Figure 13.2 . Type **AROUT** in the Values text box.
5. Click Execute.

A new window is opened with the following code:

```
DECLARE @RC int
DECLARE @CustomerID nchar(5)
SELECT @CustomerID = N'AROUT'
EXEC @RC = [Northwind].[dbo].[CustOrderHist] @CustomerID
DECLARE @PrnLine varchar(8000)
PRINT 'Stored Procedure: Northwind.dbo.CustOrderHist'
SELECT @PrnLine = ' Return Code = ' + CONVERT(varchar, @RC)
PRINT @PrnLine
```

If, instead of selecting Open, you selected Script to New Window as… and then Execute, the call to the stored procedure would be generated with just the variable declarations and calling syntax. The dialog box in Figure 13.2 would not be displayed, and the variable assignments would need to be scripted manually.

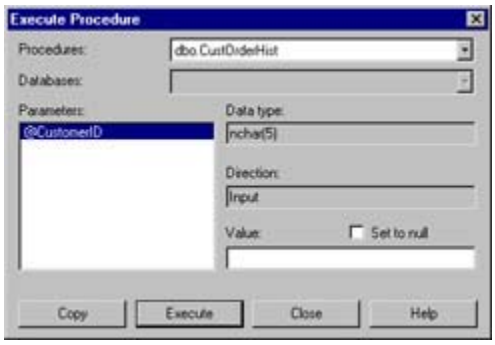


Figure 13.2: Execute Procedure parameter entry.

Below the database objects in the Object Browser window is a Common Objects folder listing the SQL Server built-in functions grouped by purpose. Below each of these functions is a Parameters folder that lists the data types of the parameters. You can automatically script the template for executing the function, similar to automatically scripting a stored procedure.

The Object Browser works well if you know what you are looking for. If you have an idea of the name of the object you are searching for but not which database it is in, finding it could be troublesome. Instead of using the Object Browser in this case, you can search for an object by name in the Object Search across all databases on your server.

As an example, let's search for all objects that have customer-related names. These steps assume that you are connected to the Northwind database in Query Analyzer. However, the particular database you are connected to does not matter. By default, the search is done over all databases on the server when you follow these steps:

1. Click the Tools menu and select Object Search or press f4 .
2. In the Object Name drop-down box, type ***cust*** . The asterisks are treated as wildcards. The search will find all objects that have *cust* in the name.
3. Check the All object types check box. By default, only table names are searched.
4. Click Find Now.

In addition to automatically scripting various tasks related to database objects, the Object Browser makes available a series of template scripts for writing common tasks. You can view these templates by clicking the Templates tab in on the bottom of the Object Browser. This feature is particularly useful when you are not familiar with the syntax for a required task. For example, there is a template for creating an INSTEAD OF trigger in the Create Triggers folder. Cursors can be somewhat difficult to use. A few templates are provided under the Using Cursor folder.

SQL Debugger

If you are using SQL Server as a database back end, chances are fairly good that you are calling stored procedures. At some point during the development process, you will probably trace an error to the database. You might have used the data environment of

Visual Basic and the T-SQL debugger to step through the stored procedure. Setup was a bit of a hassle, and stepping though the stored procedure was rather a rather slow process. Now Query Analyzer has a built-in debugger that addresses most of the previous usability problems. It is installed by default under the development tools in the SQL Server setup.

In the Northwind database, we will create a new stored procedure containing an error and step through it to find our mistake. Let's assume that one of the managers at the fictitious Northwind company requested the ability to look up the number of orders an employee has placed by full name. We create the following stored procedure to handle the request:

```
CREATE PROC EmployeeOrderCount(@sFirstName nvarchar(10),
             @sLastName nvarchar(20))
AS
DECLARE @iEmployeeID int

SELECT @iEmployeeID FROM Employees
WHERE FirstName = @sFirstName AND LastName = @sLastName

SELECT COUNT(*) FROM Orders
WHERE EmployeeID = @iEmployeeID
GO
```

There are better ways to go about the requested task. An INNER JOIN between the Employees and Orders tables would produce the desired results, but let's use this stored procedure anyway for our debugging walk-through. We test the stored procedure using an employee name listed in the Employees table with the following line:

```
EmployeeOrderCount 'Robert', 'King'
```

And we find that no rowset is returned. Now we will debug the stored procedure and find the cause of these incorrect results.

1. Find the EmployeeOrderCount stored procedure in the Object Browser. You might have to right-click the stored procedure folder and select Refresh from the context menu before the new stored procedure is displayed.
2. Right-click the EmployeeOrderCount stored procedure and select Debug.
3. A window like the one in Figure 13.2 is displayed. Type **Robert** in the Value text box.
4. Click the @sLastName item in the Parameters list and type **King** in the Value text box.
5. Click Execute, and you will see the debugger (shown in Figure 13.3) with the contents of your stored procedure.

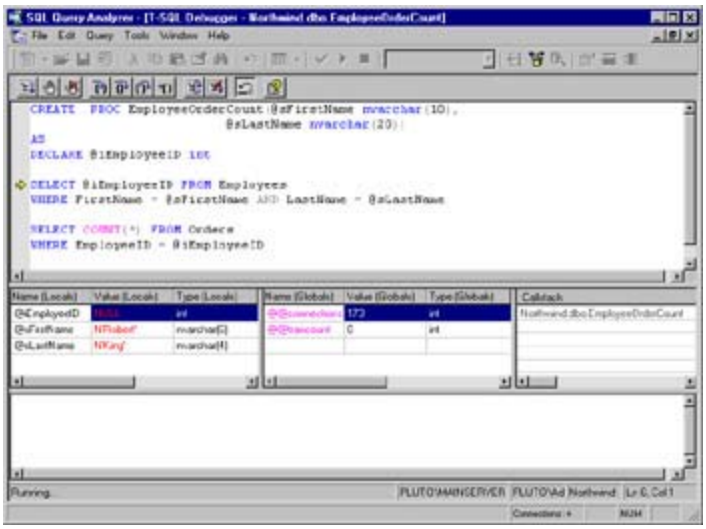


Figure 13.3: The SQL Debugger.

6. The yellow arrow indicates the current line of execution. Click the fourth button from the left on the debugger toolbar or press f10 to step through this line of code.
7. Take a look at the Locals watch window. The @iEmployeeID will still be NULL. By this point, we should have a value because we passed in the name of an existing employee. There is an error in the first SELECT statement.
8. Press the first button on the left or f5 to continue executing the stored procedure.
9. Close the SQL Debugger.

Now we know where the problem lies and we can correct it. The SELECT statement needs to be changed to read the employee ID into the @iEmployeeID local variable. The following ALTER PROCEDURE statement makes this correction:

```
ALTER PROC EmployeeOrderCount(@sFirstName nvarchar(10),
                    @sLastName nvarchar(20))
AS
DECLARE @iEmployeeID int

SELECT @iEmployeeID = EmployeeID FROM Employees
WHERE FirstName = @sFirstName AND LastName = @sLastName

SELECT COUNT(*) FROM Orders
WHERE EmployeeID = @iEmployeeID
```

After executing this script to make our changes, we go through the same steps to see the properly working stored procedure. Notice that the @iEmployeeID value changes after we step through the first select statement.

The SQL Debugger has a few windows displaying information. The Locals window displays the local variables declared in the stored procedure as well as the parameters. The Globals watch window displays the SQL Server global variables. All global variables in SQL Server start with the @@ symbols. You change the values of the local variables, but not the global variables, in the Local watch window. The Callstack window

is useful for debugging stored procedures that call other stored procedures. It displays the order in which the stored procedures are invoked. The bottom window simply displays the output of the stored procedure.

There are some features common to debugging in Visual Basic or Visual C++ that you will not see here. You cannot set breakpoints or create conditional watch statements based on custom expressions.

Tip When debugging a stored procedure that generates an error, add the @@ERROR variable to the global watch window. As you step through each line, monitor the @@ERROR value. As soon as you see the variable change to a nonzero value, you'll know that the error occurred on the line you just executed.

User-Defined Functions

User-defined functions are similar to stored procedures in that they can return rowsets and they can be executed in the same fashion. This, however, is where the similarity between stored procedures and user-defined functions ends. In this section, we cover the three function types: scalar, table, and inline. User-defined function definitions require an explicit RETURN statement that terminates the function. BEGIN and END statements encapsulate the function body, unless it is an inline function. The standard SQL Server data types, such as *int, bigint,* and *sql_variant,* can be used as return values . The *table* data type is particularly powerful when used as the return value. You can write a view as a user-defined function that takes a parameter to limit the length of the rowset returned. User-defined data types can be return values as well. If a text column is part of the returned table, only the first 256 characters will be returned. There are three types of user-defined functions: scalar, table, and inline. Let's look at them more closely.

Scalar User-Defined Functions

Simply put, *scalar user-defined functions* return a variable of any data type other than *table* . We can rewrite the EmployeeOrderCount stored procedure example we examined earlier in this chapter as a user-defined function to return the order count for an employee directly as a variable rather than in a rowset.

```
CREATE FUNCTION fn_EmployeeOrderCount(@sFirstName nvarchar(10),
                  @sLastName nvarchar(20))
RETURNS int
BEGIN
DECLARE @iEmployeeID int
DECLARE @iOrderCount int

SELECT @iEmployeeID = EmployeeID FROM Employees
WHERE FirstName = @sFirstName AND LastName = @sLastName

SELECT @iOrderCount = COUNT(*) FROM Orders
WHERE EmployeeID = @iEmployeeID

RETURN(@iOrderCount)
```

```
END
```

This function can be called as follows:

```
SELECT dbo.fn_EmployeeOrderCount('Robert', 'King')
```

Scalar user-defined functions can be used in the selection list but not as the subject of a FROM clause. Only user-defined functions that return tables can be used in the FROM clause. In addition, scalar user-defined functions must be qualified with the owner name; otherwise, an error will be generated, indicating that the function name is unrecognized.

Table User-Defined Functions

Views are good for hiding the underlying data structure, applying access security, and joining separate tables into a single rowset. You have some control over filtering the data in a WHERE clause, but there is no room for conditional logic. *Table user-defined functions* —user-defined functions that return tables—can incorporate conditional logic. Furthermore, views cannot internally use the ORDER BY clause. If you want to specify a sort order for a view, it must be done in a SELECT statement against the view. An ORDER BY clause can be specified in a table user-defined function but not in an inline user-defined function.

There are a few differences in the declaration of table user-defined functions. When the *table* data type is the return type, the variable name immediately follows the RETURNS statement. The table structure is explicitly defined in the function header. You'll also notice that the variable is not explicitly returned.

Let's rewrite the Customer and Supplier by City view in the Northwind database as a function that takes a specific city name:

```
CREATE FUNCTION fn_CustomerSupplierByCity(@sCity varchar(34))
RETURNS @tCutomerSupplier table
(City nvarchar(15) NULL,
 CompanyName nvarchar(40) NOT NULL,
 ContactName nvarchar(30) NULL,
 Relationship varchar(9) NOT NULL)
AS
BEGIN

INSERT INTO @tCutomerSupplier
SELECT City, CompanyName, ContactName, 'Customers' AS Relationship
FROM Customers
UNION SELECT City, CompanyName, ContactName, 'Suppliers'
FROM Suppliers
WHERE City = @sCity
ORDER BY CompanyName

RETURN
END
```

This user-defined function returns a table and cannot be called in the column listing of a SELECT clause. Rather, it can be used as the subject of a FROM clause as follows:

```
SELECT * FROM fn_CustomerSupplierByCity('London')
```

Inline User-Defined Functions

Inline user-defined functions are a shortened version of table user-defined functions. Table user-defined functions can contain multiple statements, whereas inline user-defined functions contain a single SELECT statement as the return value. There is no function body; therefore, there is no BEGIN…END pair. The table definition is obtained from the structure of the SELECT statement result.

The syntax for creating this type of function is more compact, but there are additional restrictions. Interestingly, as in a view, the ORDER BY clause cannot be used within the definition. All tables must be qualified with the owner name.

Inline user-defined functions are dependent on the tables referenced in the SELECT statement to provide the underlying schema. This could cause a problem if the referenced tables are changed. It is clear that if, for example, you drop a referenced table, the function will no longer work.

The WITH SCHEMABINDING option used at the creation of the function restricts the referenced tables from being dropped or altered until the option is removed. It can also be applied to views, which you will see later when we cover indexed views. The function or view must meet certain requirements before it can be schema bound. The referenced tables must be fully qualified with the owner name. In addition, the columns in the SELECT statement must be specified. SELECT * is not permitted. The creator of the function or view must own the tables. Otherwise, the table owner must grant the REFERENCES permission on the referenced table to the user creating the function or view.

Note Schema binding is not enforced across functions or views comprising tables in multiple databases. All schema-bound objects must reside in the same database.

Let's rewrite the fn_CustomerSupplierByCity we created earlier as an inline user-defined function.

```
CREATE FUNCTION fn_CustomerSupplierByCity_sb(@sCity varchar(34))
RETURNS table
WITH SCHEMABINDING
RETURN
SELECT City, CompanyName, ContactName, 'Customers' AS Relationship
FROM dbo.Customers
UNION SELECT City, CompanyName, ContactName, 'Suppliers'
FROM dbo.Suppliers
WHERE City = @sCity
```

You'll notice that a few changes are made to the SELECT statement. The ORDER BY statement is not permitted in an inline user-defined function. Since the WITH SCHEMABINDING option was used, SQL Server will generate an error if someone attempts to drop the Customers or Suppliers table.

Note The WITH SCHEMABINDING option can also be used in the creation of views. SQL Server will enforce the same restrictions on views as it does with inline user-defined functions using SCHEMABINDING.

Referential Integrity Enhancements

Referential integrity guarantees that tables containing related information maintain their relationships. For example, in the Northwind database, the Orders table contains the shipping information for an order, and the Order Details table lists the products ordered. You cannot delete a row in the Orders table until you delete all the related rows in the Order Details table. Previously, if you wanted to delete an order without having to first delete the order detail information, you could use a delete trigger on the Orders table that would delete the related rows in the Order Details table. You could also use a stored procedure. Cascading updates and deletes provide an efficient alternative that is easier to implement.

Cascading Updates and Deletes

Foreign key constraints currently enforce table relationships. They prevent data from changing when a DELETE or an UPDATE statement would change the relationship between tables. This idea has been extended to incorporate cascading updates and deletes as defined in the ANSI SQL standard. The default definition of a foreign key is equivalent to using the NO ACTION constraint as follows:

```
ALTER TABLE dbo.[Order Details] ADD
CONSTRAINT FK_Order_Details_Orders FOREIGN KEY
   (OrderID) REFERENCES Orders (OrderID)
   ON DELETE NO ACTION
   ON UPDATE NO ACTION
```

This indicates that if a delete or an update should affect the OrderID column in a row of the referenced Orders table, no action is to be taken. Therefore, the following statement will fail:

```
DELETE FROM Orders where OrderID=10250
```

We can change this foreign key definition to automatically delete rows in the Order Details table when the related order is deleted. First we must drop the constraint, then recreate it with the CASCADE option:

```
ALTER TABLE dbo.[Order Details]
   DROP CONSTRAINT FK_Order_Details_Orders
GO
```

```
ALTER TABLE dbo.[Order Details] ADD
CONSTRAINT FK_Order_Details_Orders FOREIGN KEY
   (OrderID) REFERENCES Orders (OrderID)
   ON DELETE CASCADE
   ON UPDATE CASCADE
```

Now, if we execute the DELETE statement again, the related rows in the Order Details table will be deleted as well.

Trigger Enhancements

A *trigger* is a special type of stored procedure that executes when an insert, update, or delete occurs against a table. Triggers are designed to fire after the operation against the table executes. For example, a DELETE trigger executes after the row(s) targeted for the delete have been deleted. If the rows cannot be deleted—say, due to a foreign key constraint—the trigger is never fired. These are now known as AFTER triggers, since they occur after the triggering event. SQL Server 2000 offers INSTEAD OF triggers that execute in place of the INSERT, UPDATE, or DELETE operation.

INSTEAD OF

SQL Server 2000 provides a new INSTEAD OF trigger type that fires in place of the intended action. An INSTEAD OF trigger gives you much greater control over table data interaction than do standard triggers. For example, INSTEAD OF gives you the ability to selectively accept or reject rows in a large batch insert.

The most powerful benefit of the INSTEAD OF trigger is its ability to be added to views. In prior versions of SQL Server, you could update views if only one table was the subject of the update, even if multiple tables were used in creating the view. You were out of luck, however, if you wanted the update to affect more than one table in the view. INSTEAD OF triggers remove this limitation. Now you can update multiple tables through a view by handling the change with an INSTEAD OF trigger. So, you can update and insert into the same tables you select from. This feature also adds a level of security by restricting the rows a user is permitted to update.

The Products by Category view in the Northwind database gives us a listing of categorized products. We can add products and categories in separate INSERT statements, but we cannot insert into the two tables using the same statement. The following INSERT statement produces an error:

```
INSERT INTO [Products by Category] (CategoryName, ProductName,
   QuantityPerUnit, UnitsInStock, Discontinued)
VALUES ('Beer', 'Outback Amber', '24 - 355 ml bottles', 10, 0)
```

Let’s create an INSTEAD OF trigger that will allow us to insert into both the Products and the Categories tables:

```
CREATE TRIGGER tr_Products_By_Category_Insert_Insead_Of
```

```
ON [Products By Category] INSTEAD OF INSERT
AS
BEGIN

— Insert an unmatched category into the Categories table
INSERT INTO Categories (CategoryName)
SELECT inserted.CategoryName FROM
inserted LEFT JOIN Categories ON
inserted.CategoryName = Categories.CategoryName
WHERE Categories.CategoryName IS NULL

— Update products if the product name appears in the Product table
UPDATE Products SET Products.CategoryID = Categories.CategoryID,
   QuantityPerUnit = inserted.QuantityPerUnit,
   UnitsInStock = inserted.UnitsInStock,
   Discontinued = inserted.Discontinued
FROM inserted INNER JOIN Categories ON
inserted.CategoryName = Categories.CategoryName
          INNER JOIN Products ON
inserted.ProductName = Products.ProductName

— Insert the Product if the product name does not
— appear in the Product table
INSERT Products (CategoryID, ProductName, QuantityPerUnit,
           UnitsInStock, Discontinued)
SELECT Categories.CategoryID, inserted.ProductName,
       inserted.QuantityPerUnit, inserted.UnitsInStock,
       inserted.Discontinued
FROM inserted INNER JOIN Categories ON
inserted.CategoryName = Categories.CategoryName
          LEFT JOIN Products ON
inserted.ProductName = Products.ProductName
WHERE Products.ProductName IS NULL



END
GO
```

Now try the prior INSERT statement. It will change both the Products and the Categories tables. The trigger still has access to the inserted and deleted tables, which work the same way as they do in standard triggers. This example shows that the inserted table can still be used in SELECT statements. The affected tables behave differently, though. In older triggers, the Products and Categories tables would closely match the trigger-specific inserted table. In other words, the affected tables would already contain the new data, as though the insert or update had already occurred. Conversely, in INSTEAD OF triggers, the underlying tables contain the same data as they would prior to the insert or update. Therefore, the Products and Categories tables remain unaffected until the INSERT and UPDATE statements fire.

Only one INSTEAD OF trigger can be assigned per operation. Because it fires before any operation is actually performed on the table, it fires before any AFTER triggers. One of

the implications of this order of precedence is that an AFTER trigger will not fire if the triggering operation is canceled. For example, let's say that there is an INSTEAD OF trigger on the Orders table that fires on an INSERT statement as well as a corresponding AFTER trigger on the same table. If the INSTEAD OF trigger cancels the insert, the AFTER trigger does not fire. If the INSTEAD OF trigger performs an operation different from the one that fired it, any AFTER triggers associated with the new operation fire. For example, if an INSTEAD OF trigger fires on an INSERT statement but performs an update, any subsequent AFTER triggers associated with an update fire.

Bulk insert statements also support INSTEAD OF triggers when updating views. This is useful for managing legacy systems. By default, bulk insert does not fire triggers. You have to explicitly specify that triggers should be fired:

```
BULK INSERT [Products By Category]
FROM 'c:\import.txt'
WITH (FIRE_TRIGGERS, datafiletype='char', fieldterminator=',')
```

Create an IMPORT.TXT file containing the following text and save it to the root of your C: drive before executing the BULK INSERT example:

```
Seafood,Swordfish Steak,16 pieces,10,0
Condiments,Cajun Mustard,24 - 8 oz jars,32,0
Produce,Andouille Sausage,1 kg pkg.,8,0
```

AFTER

As the name implies, AFTER triggers occur after the triggering operation has already completed. Unlike INSTEAD OF triggers, AFTER triggers can be created only against tables. If you used triggers in SQL Server 7.0, you are already familiar with AFTER triggers. Unless otherwise specified, a trigger falls into this category. The following declarations may have different syntax, but they have the same effect:

```
CREATE TRIGGER tr_testTrigger_insert ON testable
FOR DELETE
AS …

CREATE TRIGGER tr_testTrigger_insert ON testable
AFTER DELETE
AS …
```

Previously, you could have multiple triggers fired by the same operation on the same table, but you couldn't control their order. A new stored procedure, sp_settriggerorder, allows you to specify the firing sequence. You can specify the first and last triggers fired, but you cannot specify the order of the rest of the triggers. Therefore, if you have four or more triggers, the order of precedence cannot be determined.

Suppose that a trigger is designated as the first in the firing sequence. Another trigger is set to fire last. The two remaining triggers will fire in an undetermined sequence, and you could get undesired results.

As you'd expect, you cannot have more than one trigger set as the first to execute. The same applies to the last. Furthermore, the first trigger to fire must be different from the last one fired. The following statement sets tr_testTrigger_insert as the first trigger to be executed:

```
sp_settriggerorder 'tr_testTrigger_insert', @order='first'
```

Warning If a trigger is altered, the order of precedence is lost. It must be reset using sp_settriggerorder.

Indexed Views

A *view* is a predefined query that you can access as you would a table. It prevents users from seeing unauthorized information and hides the data structure. However, views do not give much of a performance gain. The query is precompiled for faster execution, but the tables involved in the query are scanned each time the view is used. Any aggregations and joins are established at runtime.

Now, with SQL Server 2000, you can place clustered and nonclustered indexes directly on views. Rules applying to placing indexes on views are similar to those that apply to placing indexes on tables. Only one unique clustered index can be placed on a view, just as with a table. A unique clustered index must be defined on a view before nonclustered indexes can be added. Indexes speed retrieval of the data, but just as with tables, they add additional overhead, so they should be applied carefully.

As with tables, you should determine which columns would benefit most from an index. Look for columns used in joins and aggregations. The Index Tuning Wizard suggests which columns are best suited for an index. You can also check the execution plan of the view in Query Analyzer to detect potential indexes.

A number of restrictions govern which views are eligible for an index. First and foremost, the view must be created with schema binding. (See the earlier section on INSTEAD OF triggers for more information about schema binding.) In short, schema binding prevents the underlying tables from being dropped or altered. The following settings must be configured in order for the index to be created:

```
SET ARITHABORT ON
SET CONCAT_NULL_YIELDS_NULL ON
SET QUOTED_IDENTIFIERS ON
SET ANSI_NULLS ON
SET ANSI_PADDING ON
SET ANSI_WARNINGS ON
SET NUMERIC_ROUNDABORT OFF
```

By default, Query Analyzer has all these settings except one. The ARITHABORT option is defaulted to OFF. You can get a list of your session settings by executing DBCC USEROPTIONS. If you are interested in checking just one setting, you can use the new SESSIONPROPERTY function, which returns 1 if true and 0 if false, as follows:

```
SELECT SESSIONPROPERTY('ARITHABORT')
```

All functions in the view must be deterministic. Deterministic functions, given the same parameters, always return the same result. SQRT is one such function. SQRT(4) always returns 2, no matter how many times you execute it. In contrast, RAND clearly returns a different value each time. USER_NAME is another example of a nondeterministic function. Its return value is different depending on the connected user. The @@servername is also nondeterministic. It returns a different value depending on the server.

If a GROUP BY is used in the function, the new COUNT_BIG function must be included in the select list. This function was covered in the earlier section about the new data types.

The remaining restrictions are as follows:

- No OPENROWSET or OPENQUERY functions
- No views can be referenced with the indexed view
- No MAX or MIN functions
- No AVG aggregate function
- No STDEV or VARIANCE statistical functions
- No UNIONS
- No binary columns such as *text, ntext,* or *image*
- No COUNT(*) or COUNT(<column list>) functions
- No DISTINCT clauses
- No OUTER joins
- No subqueries
- No self-joins
- No COMPUTE or COMPUTE BY
- No HAVING, ROLLUP, or CUBE following a GROUP BY clause
- No full-text searches (for example, CONTAINS or FREETEXT)
- No TOP
- No SUM on a nullable column or expression

These are a lot of restrictions to keep in mind. You can check if a prospective view meets these qualifications with the new IsIndexable property in the ObjectProperty function. Like the SESSIONPROPERTY function, a return value of 1 indicates true and 0 indicates false:

```
SELECT ObjectProperty(object_id('Order Subtotals'), 'IsIndexable')
```

Tip You might not be able to use the AVG function in an indexed view, but there is a work-around. You can still use the SUM function, and the COUNT_BIG aggregate is required if the GROUP BY is used. You can get the average by dividing the SUM function by the COUNT_BIG aggregate.

Despite these restrictions, indexed views offer a powerful advantage. The end user of an indexed view can use it like a normal view. Every time the base tables are updated, the

index is updated. We will examine how the Northwind database can benefit from an indexed view.

The Products by Category view in the Northwind database joins the Products table to the Category table by the ProductID. We can look at the execution plan of this view in Query Analyzer:

```
CREATE VIEW EmployeeSales
WITH SCHEMABINDING
AS
SELECT dbo.Employees.EmployeeID, FirstName, LastName,
   SUM(UnitPrice*Quantity) AS SaleTotal, COUNT_BIG(*) as SaleCount
FROM dbo.Employees INNER JOIN dbo.Orders
          INNER JOIN dbo.[Order Details]
ON dbo.Orders.OrderID = dbo.[Order Details].OrderID
ON dbo.Employees.EmployeeID = dbo.Orders.EmployeeID
GROUP BY dbo.Employees.EmployeeID, FirstName, LastName
```

The EmployeeID columns in the Employees and Orders tables are used in a join. This makes it a good candidate for an index:

```
CREATE UNIQUE CLUSTERED INDEX idx_EmployeeSales
ON [EmployeeSales](EmployeeID)
```

Tip When you place an index on a column in a view, you can then make sure the index is used by placing the view in a user-defined function, the parameters of which match the indexed columns. For example, in order to guarantee that the index in the EmployeeSales table is used, you could create a user-defined function that uses the NOEXPAND hint and takes the EmployeeID as a parameter.

The EmployeeSales view is grouped by employee information, so we set up the index based on the Employee ID. This will significantly speed up our data retrieval. The goal here is to reduce the number of reads SQL Server uses to get our data. A logical read comes from cached data residing in memory; a physical read goes against the hard disk. Since SQL Server might have already cached the information we are retrieving in this test case, we'll examine the number of logical reads SQL Server does when using this indexed view.

SQL Server includes a new query hint, OPTION (EXPAND VIEWS), which forces SQL Server to expand all the underlying SELECT statements that make up a view and ignore all indexes. Another query hint, NOEXPAND, does the opposite. It forces SQL Server to use the indexes on the view but only if the view appears in the FROM clause of the SELECT statement. The query optimizer determines the best method of execution for a SQL statement. It if determines that there is no significant gain in using the indexed view, it uses the indexes of the base tables.

Let's examine the difference in our EmployeeSales when the optimizer uses the index and when it doesn't. Turning the STATISTICS IO option on displays the physical reads and writes to the hard drive. Turning the STATISTICS TIME option on displays the

execution time. The following script forces SQL Server to ignore any indexes on the view:

```
SET STATISTICS IO ON
SET STATISTICS TIME ON

SELECT * FROM EmployeeSales
OPTION (EXPAND VIEWS)
```

At the end of the result statement, you will get the following output (or something close to it):

```
Table 'Order Details'. Scan count 829, logical reads 1670, physical
reads
0, read-ahead reads 0.
Table 'Orders'. Scan count 9, logical reads 19, physical reads 0,
read-ahead reads 0.
Table 'Employees'. Scan count 1, logical reads 2, physical reads 0,
read-ahead reads 0.

SQL Server Execution Times:
   CPU time = 19 ms, elapsed time = 19 ms.
```

The SELECT statement results in a total of 1691 logical reads and 19 milli-seconds of execution time. Your results might vary slightly. These tests were conducted on a PC Pentium running Windows 2000 Advanced Server with a 700MHz Pentium III and 128MB of RAM.

Now let's execute this same statement but with the NOEXPAND hit to force it to use the index:

```
SET STATISTICS IO ON
SET STATISTICS TIME ON

SELECT * FROM EmployeeSales
WITH (NOEXPAND)
```

Much less output is generated at the end of the result set:

```
Table 'EmployeeSales'. Scan count 1, logical reads 2, physical reads 2,
read-ahead reads 0.

SQL Server Execution Times:
   CPU time = 8 ms, elapsed time = 8 ms.
```

The EmployeeSales view was scanned with only two logical reads. That's quite a reduction from 1691 logical reads. The execution time is more than cut in half, from 19 ms. to 8 ms. This is an excellent example of how indexes on views can vastly improve performance. The increase is certainly worth working around the copious restrictions.

Meta Data Services

Strictly speaking, metadata is data about data. In this case, the term *metadata* refers to information about a company's information technology assets, spanning everything from coding standards to data warehouses to reusable objects and naming conventions. This is more commonly known as a *repository* , which is a tool for storing, cataloging, and sharing information about the various pieces of a corporation's IT resources and how they interrelate. If this idea sounds rather abstract, you can think of it as a giant enterprisewide card catalog of technical resources. By providing a common format and a central location for storing such information, developers can easily share metadata about a company's technical information.

In December 1996, Microsoft released the Microsoft Repository 1.0 software development kit (SDK). This SDK was, and still is, composed of two major components: a COM interface and the repository engine. Developers can use the COM interface to access and manage the repository. They adhere to the Open Information Model (OIM) standard, which this chapter will cover in more depth later.

The repository engine is the underlying storage mechanism for these information models. It can communicate with both a Microsoft Access database and a SQL Server database. It originally used a Microsoft Access database to store the repository information, which might work for a department but does not scale well to meet the demands of an enterprise.

Later, Microsoft Repository 2.1 shipped with SQL Server 7.0, which could meet such scalability demands. Under the hood, the repository itself is stored in the MSDB database. The tables that are part of repository start with Rtbl, Dbm, Dtm, Dts, Gen, Mds, Ocl, Olp, Tfm, Uml, and Umx. The stored procedures that are part of the repository start with r_i. This database was closely coupled with SQL Server to facilitate storage of Data Transformation Services (DTS) packages and database schema. If you wanted to store additional types of information, such as Visual Basic code, you had to use other utilities to access the repository, such as the Visual Component Manager or the Repository Browser, which comes with the Microsoft Repository SDK. (For more information, you can visit http://msdn.microsoft.com/repository.)

As the Microsoft Repository evolved, it became more and more closely tied to SQL Server. Now, with SQL Server 2000, you can directly browse the contents of the repository through Enterprise Manager in the Contents folder of the Meta Data Services. If you have any DTS packages stored in the Meta Data, you might notice a Microsoft Data Warehousing Framework item in the Contents folder. This folder contains the DTS package. All users connecting to the database will see the same items under this folder.

The Meta Data Services provide three browsing options: End User, Power User, and Administrator. In the End User Browse mode, you can make no changes; you can only browse information. You can create, edit, and delete objects as well as edit object properties in the Power User Browse mode. The Administrator Browse mode can do everything the Power User Browse mode can do and more. Only the Administrator Browse mode can view the Information Models folder. The Information Models define

how an object is represented in the Meta Data Services. For example, the DTS information model defines how DTS packages are represented in the repository. Administrators can also import new Information modes. In addition, only Administrators can view repository identifiers such as the ObjID and InternalID, which are normally hidden object properties.

Within Enterprise Manager, you can view the data in the repository only from the End User Browse mode. This is the default view and cannot be changed. The Meta Data Services can be browsed in Stand Alone mode, which allows you to change Browse modes. For ease of administration, you can load the Meta Data Services into a Microsoft Management Console (MMC) and keep it separate from Enterprise Manager:

1. From your Windows Start menu, select Run.
2. Type **MMC** and click OK.
3. From the main console toolbar, select Add/Remove Snap-in…
4. On the Add/Remove Snap-in dialog box, click Add.
5. Scroll down and select Meta Data Services. Click Add.
6. In the description section of the Add/Remove Snap-in dialog box, you should now see "Microsoft Meta Data Browser." Click OK.
7. Right-click the Meta Data Services node and select Register Database.
8. Enter your server name and the proper administrator username and password.
9. Select the MSDB database from the database drop-down list.
10. Change the Browse mode from End User to Administrator. Click OK.
11. Click the registered database node and you'll see the Contents and Information Models nodes.
12. Click the Contents node for a list of items in the Meta Data Services.
13. Click the Console menu in the main MMC window and select Save.
14. Enter the filename **Meta Data Services** and click Save.
15. Click the Console menu in the main MMC window and select Exit.
16. From your Windows Start menu, select Program Files | Administrative Tools. You'll notice that the Meta Data Services is now accessible through this menu.

Transact-SQL

E. F. Codd developed Structured Query Language (SQL) in the mid-1970s for IBM. SQL was originally called Structured English Query Language (SEQUEL) until IBM shortened the name due to legal problems. It was designed to work with System R, an early prototype of a relational data-base management system (RDBMS). Although IBM produced SQL, Oracle was the first to bring it to market in 1979. In 1989, ANSI attempted to standardize the language so that the same SQL commands could run against databases from multiple vendors. Later, in 1992, it released an update to this standard, known as SQL 92. Since then a number of vendors have implemented their own versions of SQL to address certain needs, such as creating indexes and conditional logic. In addition, as with many standards in the information technology industry, vendors do not implement all the standard's recommendations. For example, cascading updates and deletes are part of the SQL 92 standard but were not implement in SQL Server until this latest release.

SQL, simply put, is a scripting language designed to provide a standard means of creating database objects, managing data, and administering security. These three operational categories are defined as Data Definition Language (DDL), Data Manipulation Language (DML), and Data Control Language (DCL). DDL is used to create, drop, define, and alter database objects such as tables and views. CREATE TABLE falls into this category. DML is used to access database objects. SELECT, INSERT, UPDATE, and DELETE fall into this category. DCL is used to manage database object security. GRANT and REVOKE are examples of DCL. We will be exploring these three categories in more depth as they are implemented in SQL Server 2000.

Transact-SQL (T-SQL) is a vendor-specific implementation, originally developed through a partnership between Microsoft and Sybase in 1987. As such, the implementations of T-SQL in the early versions of Sybase's and Microsoft's SQL Server were very similar. By 1994, the partnership ended, and the database management systems have since diverged.

T-SQL is an extension to SQL that offers additional data types, conditional logic, temporary objects, error handling, standard system stored procedures, and more. In addition, like most database vendors, not all SQL 92 specifications are implemented. Consider the following SQL statement:

```
DROP TABLE dbo.Employees RESTRICT
```

It indicates that the Employee table should not be dropped. If a drop is attempted, an error should be thrown as though it were subject to schema binding. It is a valid SQL statement according to the SQL 92 specification, but it does not work in SQL Server 2000.

You have already seen some T-SQL-specific functionality in some of the preceding examples in this chapter. The COUNT_BIG and SQL_VARIANT_PROPERTY functions are both unique to T-SQL. These functions are designed to work with certain

SQL Server 2000 enhancements. In addition, systemwide stored procedures are also part of T-SQL. The SP_HELP and SP_HELPTEXT functions can be used to give you information about various database objects. Using SP_HELP with a view generates a listing of the columns and data types the view returns; SP_HELPTEXT returns the SELECT statements of the view itself. T-SQL further expands on the SQL 92 standard, providing condition logic, which makes it approach the look of higher-level languages. The IF statement provides conditional logic, and the WHILE..WEND pair provides a looping construct. The RAISERROR statement throws an error to the caller. You can use these features in combination with each other to implement business logic directly in the database. The following stored procedure prevents orders over three years old from being deleted. If the OrderID is null or not specified, it will return an error:

```
ALTER PROC deleteOrder(@iOrderID int)
AS
DECLARE @iYearAge int

IF @iOrderID IS NULL
  RAISERROR ('OrderID cannot be NULL', 16, 1)
ELSE IF NOT EXISTS(SELECT OrderID FROM Orders
                    WHERE OrderID = @iOrderID)
  RAISERROR ('OrderID %i not found', 16, 1, @iOrderID)
ELSE
  BEGIN
    SELECT @iYearAge = DATEDIFF(year, OrderDate, GETDATE())
    FROM Orders WHERE
    OrderID = @iOrderID
IF @iYearAge>3
      DELETE Orders WHERE
      OrderID=@iOrderID
   ELSE
     RAISERROR('OrderID %i < 3 years old', 16, 1, @iOrderID)
 END
GO
```

Data Definition Language

DDL is used to create and alter database objects. You can use the Enterprise Manager to create your database objects. However, if you familiarize yourself with these statements, you can take that knowledge to other databases without graphical tools. In addition, you will become better acquainted with the available options.

The three major commands comprising DDL are CREATE, DROP, and ALTER. You can apply these commands to most of the objects in your database. For example, you can CREATE, DROP, and ALTER tables, views, functions, stored procedures, triggers, and databases. Other objects such as indexes and defaults can be created and dropped but not altered. Constraints are restrictions placed on a table. They help stop people from compromising the integrity of your data. Primary and foreign keys fall into this category, along with defaults and rules.

Let's say that the fictitious Northwind company adds a new employee with a last name longer than 20 characters. The current LastName column in the Employees table would not be able to hold it. The following ALTER TABLE statement expands that column:

```
ALTER TABLE Employees
ALTER COLUMN LastName nvarchar(40) NOT NULL
```

If you created the EmployeeSales view from an earlier sample, an error will be generated notifying you that one or more objects depend on this column. That's because the EmployeeSales view is schema bound to the Employees table. If a table is referenced by a view declared with SCHEMABINDING, the view will need to be altered without that option. In this example, if the EmployeeSales view is recreated with an ALTER VIEW statement that does not have the WITH SCHEMABINDING option, this ALTER TABLE statement will work. It's better to alter the view rather than recreate it, since the ALTER statement will preserve security settings. Otherwise, if it is dropped and recreated, you will have to re-establish the security rights associated with the view.

Data Manipulation Language

Most developers are familiar with the DML portion of SQL. The SELECT, INSERT, UPDATE, and DELETE statements are used to create, manipulate, and delete data within a relational database. The READTEXT, UPDATETEXT, and WRITETEXT statements are used to handle binary text and image data.

The SELECT statement is the cornerstone of your data retrieval. It can be used to join multiple tables and present them in a uniform result set. The following statement joins the Territories and Region table:

```
SELECT TerritoryID, TerritoryDescription,
       RegionDescription
FROM Territories INNER JOIN Region
ON Territories.RegionID = Region.RegionID
```

The INSERT statement is used to add rows to tables and views. The following example adds a new employee to the Employees table. The table has more columns than we have values, but those columns accept null values, so we are not required to fill them in. The EmployeeID column does not accept nulls, but we are not stating an explicit value because it is an identity column. SQL Server will assign the value of the EmployeeID column:

```
INSERT Employees (LastName, FirstName, Title)
VALUES ('Gordon', 'Lee', 'Sales Representative')
```

The UPDATE statement is used to change column values in existing rows. It has the potential to update all the rows in a table unless a filter is applied in the WHERE clause. Since the update modifies data, it populates the transaction log with each row affected. On a view with multiple tables, this adds up. Be sure to take into consideration the steps the optimizer follows when altering data.

The following example uses the UPDATE statement to add some additional information to the row in the Employees table containing Lee Gordon:

```
UPDATE Employees SET BirthDate='6/2/1971', HireDate='5/4/2000'
WHERE FirstName='Lee' And LastName='Gordon'
```

Earlier, we explored the rather straightforward INNER JOIN. It presents us with rows that are found in both tables involved in the inner join. In our earlier example, if a region had no matching territories, the region would not be listed. There are some instances in which we might want to see all the records in a particular table, whether or not they have a matching row in a related table. For example, we can get a listing of employees and see how many orders they have placed, even if they have not placed any:

```
SELECT Employees.EmployeeID, LastName, FirstName,
        COUNT(OrderID) As OrderCount
FROM Employees LEFT JOIN Orders
ON Employees.EmployeeID=Orders.EmployeeID
GROUP BY Employees.EmployeeID, LastName, FirstName
ORDER BY OrderCount DESC
```

This statement uses a LEFT JOIN, which lists all rows from the table on the left side of the join, Employees, regardless of whether they have matching rows in the table on the right side, Orders. As you might have inferred, there is a RIGHT JOIN, which does the opposite. This statement also uses the ORDER BY clause to list the employees who have the most orders first. The count is listed in descending order from greatest to smallest. At the bottom, we see that Lee Gordon has not placed any orders since his hire date. If we used an INNER JOIN, he would have escaped our scrutiny. Since he has placed no orders, there are no matching rows in the Orders table.

This brings us to the DELETE statement. Like the UPDATE statement, DELETE should be used cautiously. If no filter is specified in the WHERE clause, all the rows in the table are deleted. This will also fill up the transaction log. Since Lee Gordon has not made any sales, we are forced to let him go:

```
DELETE FROM Employees WHERE
FirstName='Lee' AND LastName='Gordon'
```

In this example, we're using the full name of the employee. You should actually use the value in the EmployeeID column to perform updates and deletes. It prevents against updating or deleting rows you did not intend to affect. There could be more than one employee in the Employees table with the name John Smith, for example. If we used a DELETE statement similar to the one we just used, we would have deleted all employees with that name rather than the one intended.

When to Use ANSI SQL

If you are developing against SQL Server and have no plans to change the back-end database for your application, it makes sense to take advantage of SQL Server's specific features to make the application as fast and efficient as possible. However, this is not always the case. There might be a reason for an application to be able to work with different RDBMSs. For example, this author has worked on a commercial application that could be distributed with SQL Server, Oracle, or Informix.

In other words, you cannot take advantage of vendor-specific techniques. You also need to make a decision as to where the SQL statements are placed. Should the SQL statements be placed in stored procedures, or should they be embedded in the application code? Even the most basic SQL 92 standards are not always implemented in the same fashion. If the statements are placed in the application, the application must be aware of the type of RDBMS. This need can be removed by calling a set of stored procedures common to each implementation of your application's database. The best bet is to have a script that creates a set of stored procedures required by your application for each RDBMS. The internals of the stored procedure can vary from RDBMS to RDBMS, but the input and output will remain the same.

In order to comply with SQL 92, each vendor implements certain portions of this standard, but they also add their own extensions, creating a unique dialect of SQL. Transact-SQL (T-SQL) is one such example. You can guarantee that your SQL statements are understood as widely as possible by adhering as strictly as possible to the SQL 92 standard. You might still have to modify some of your stored procedure creation scripts when moving from RDBMS to RDBMS, but the changes will be minimized.

Both SQL Server 7.0 and SQL Server 2000 can be set up to give warnings about nonstandard SQL. You can set the Northwind database to comply with the SQL 92 standard using the following statements:

```
sp_dboption 'Northwind', 'ANSI null default', 'TRUE'
GO
sp_dboption 'Northwind', 'ANSI nulls', 'TRUE'
GO
sp_dboption 'Northwind', 'ANSI padding', 'TRUE'GO
sp_dboption 'Northwind', 'ANSI warnings', 'TRUE'
GO
sp_dboption 'Northwind', 'concat null yields null', 'TRUE'
GO
sp_dboption 'Northwind', 'cursor close on commit', 'TRUE'
GO
sp_dboption 'Northwind', 'quoted identifier', 'TRUE'
GO
```

This ensures that the database behaves in a way that is compliant with the SQL 92 standard, but it does not ensure that the SQL statements themselves comply. For example, the EXECUTE statement is valid in T-SQL, but it is not part of the SQL 92 standard. We can check compliance using the FIPS_FLAGGER option, which issues warnings but does not stop execution. You cannot set it as a databasewide option; it can be used only within a single session. It can be set to four different levels: ENTRY, FULL, INTERMEDIATE,

and OFF. Most database vendors comply with entry-level standards, so we'll use that setting. The following statements generate two warnings, telling us that both the SET and the EXECUTE statements do not meet the SQL 92 standard:

```
SET FIPS_FLAGGER 'ENTRY'
EXECUTE ('SELECT * FROM Employees')
```

Despite the options SQL Server gives for ensuring SQL 92 compatibility, there is no substitute for testing. The FIP_FLAGGER does not catch all violations; for example, it will not trap functions such as COUNT_BIG in the SQL statement. The following statements only generate a warning about using SET:

```
SET FIPS_FLAGGER 'ENTRY'
SELECT COUNT_BIG(*) FROM Employees
```

Tip During a batch process, you might have to delete the contents of an entire table. Rather than using the DELETE statement with no WHERE clause, you should use TRUNCTATE TABLE *tablename* instead. It does not fill the transaction log, so the TRUNCATE is much faster than the DELETE. In addition, since the TRUNCATE is not logged, it fires no triggers. The table cannot have any foreign keys. If you want to use the TRUNCATE TABLE statement in a batch job, you could drop the foreign keys before the truncation and recreate them afterward.

The READTEXT, WRITETEXT, and UPDATETEXT functions are designed to work the *image, text,* and *ntext* binary data types. All three of these functions require a 16-byte binary pointer to the binary data, which is returned by the TEXTPTR function. Interestingly, none of the functions uses commas to separate the parameters.

The READTEXT function is demonstrated next. It displays the contents of the Notes column in the Employees table for Anne Dodsworth by first obtaining a pointer to the binary data using a SELECT statement. Next, the READTEXT statement takes four arguments: the name of the table and column being read, the pointer value, the starting offset of where to begin the read (note the numbering starts with zero), and the number of characters to read. In this case, the number of characters is specified as zero. This indicates that 4k of data are to be read:

```
DECLARE @ptrval varbinary(16)

SELECT @ptrval = TEXTPTR(Notes)
FROM Employees
WHERE FirstName = 'Anne'
AND LastName= 'Dodsworth'

READTEXT Employees.Notes @ptrval 0 0
```

The following example makes use of WRITETEXT, which overwrites existing data. As in READTEXT, a text pointer is required. The WRITETEXT function takes three

parameters: the name of the column qualified with the table name, the binary data pointer, and the text being written.

```
DECLARE @ptrval varbinary(16)

SELECT @ptrval = TEXTPTR(Notes)
FROM Employees




WHERE FirstName = 'Anne'
AND LastName= 'Dodsworth'


WRITETEXT Employees.Notes @ptrval
'Anne has a BA degree in English from St. Augustine College. She is
fluent in French and German.'
```

The UPDATETEXT statement is more flexible than WRITETEXT. You have the option of updating only a specific portion of data rather than overwriting the entire entry. In addition, this gives the option of deleting only a segment of data by passing in an empty replacement string or inserting additional data by specifying that no bytes should be deleted. The UPDATETEXT statement takes five parameters: the name of the column qualified with the table name, the binary data pointer, the offset indicating where the replacement should go (remember the count starts at zero), the number of bytes to delete starting at the offset, and the text being written:

```
DECLARE @ptrval varbinary(16)

SELECT @ptrval = TEXTPTR(Notes)
FROM Employees
WHERE FirstName = 'Anne'
AND LastName= 'Dodsworth'

UPDATETEXT Employees.Notes @ptrval 24 7 'Literature'
```

Data Control Language

Database administrators should be familiar with the DCL aspect of SQL. The GRANT, REVOKE, and DENY statements make up DCL. SELECT, UPDATE, INSERT, and DELETE permissions can be specified on views and tables. You can get more granular and determine permissions on a column-by-column basis as well. EXECUTE permissions on stored procedures and functions can be modified using these statements as well. DCL operations such as CREATE VIEW and ALTER TABLE can also be granted and revoked to certain users.

Let’s say we don’t want to give all users access to personal employee information such as home addresses. Unknown users connect through an anonymous guest account in the

public role of the Northwind database. We can revoke access from the public group on the Employee table. Make sure you are connected to the database as a database owner:

```
REVOKE ALL ON Employees TO public
```

We still want users to obtain a list of employees, so we create a view listing only public information:

```
CREATE VIEW EmpShort
AS
SELECT EmployeeID, FirstName, LastName, Title
FROM Employees
```

SELECT access must be granted to the public group on the new EmpShort view. This will allow members of that role to query the view but not modify it:

```
GRANT SELECT ON EmpShort TO public
```

In order to test this security, we can create a couple logins and add them to the public role. You should be logged in as a database owner and defaulted to the Northwind database in Query Analyzer. The sp_addlogin stored procedure creates a login account on the SQL Server. The first parameter is the username and the second is the password, followed by the default database. The login does not have access to any databases yet. It needs to be granted access to the Northwind database. This is accomplished with the sp_adduser stored procedure, which takes the login name that was specified in the sp_addlogin statement. The next parameter is the login name that is specific to the Northwind database. You could have a login for Northwind different from the server login. Most of the time, the two will be the same. The last parameter is the role the user will play in the database:

```
EXEC sp_addlogin 'mike', 'mikepass', 'Northwind'
EXEC sp_adduser 'mike', 'mike', 'public'

EXEC sp_addlogin 'chris', 'chrispass', 'Northwind'
EXEC sp_adduser 'chris', 'chris', 'public'
```

Now you can log in as mike in Query Analyzer and attempt a SELECT against the Employees table. An error will be generated, reporting that the SELECT permission has been revoked. Instead, try using a SELECT statement against the EmpShort view. This will return a result set.

This works well if the all members of the role have uniform access. There could be a case in which you'll want to deny access to a certain member of the role without restricting access to the rest. The DENY statement fulfills this need:

```
DENY SELECT ON EmpShort TO chris
```

The user with the login chris is still a member of the public role but does not have the same level of access. Try logging in as chris and using a SELECT statement against the EmpShort table. An error will be displayed, showing that permissions have been revoked. This is particularly useful if you grant security permissions to an NT group and you want to deny access to a specific member of that group. The DENY statement does not require that the user be removed from the NT group in order to deny access to a specific member of that group.

Data Access Tools and Technologies

A developer should be familiar with the options available for accessing and using information in databases. Different techniques work in different situations. If you are importing data on a periodic basis via a batch process, you might want to take a look at the command-line utilities, which can be incorporated into DOS batch files or Windows Scripting Host scripts.

Microsoft provides a number of data access technologies for application developers. ActiveX Data Objects (ADO) allows developers to port information to the client's desktop or a Web site. OLE DB provides an opportunity to access nontraditional data sources such as Microsoft Exchange and the Active Directory as well as SQL Server or an ODBC data source. (For more information, you can visit Microsoft's Universal Data Access Web site at www.microsoft.com/data .)

Command-Line Utilities

Both the OSQL and ISQL utilities can be used to access SQL Server and perform queries from a DOS command line. However, not all utilities are created equal. The OSQL utility uses ODBC, but ISQL uses the DB-Library, which has not been updated since SQL Server 6.5. Therefore, applications that depend on DB-Library, such as ISQL, do not support some SQL Server 2000 features. For example, ISQL cannot access columns defined with the *ntext* data type and truncates any *char, varchar, nchar,* or *nvarchar* columns longer than 255 bytes. The ISQL utility was included with SQL Server 2000 for backward compatibility. You should use the OSQL utility instead, and that's what we focus on here. However, most of the things you can do with OSQL you can also do with ISQL. The following statement will not work with ISQL. The SELECT statement references the Employees table, which contains an *ntext* and an image column:

```
osql /U sa /P pass /S TESTSVR /d Northwind /Q "SELECT * FROM Employees"
```

Some of the parameters are self-explanatory. The /U indicates the username, and /P indicates the password. If you are using trusted security, you can exclude the username and password and simply put /E with nothing following. The parameter indicators are case sensitive, so /S, which indicates the server name, is not the same as /s, which takes a column separator when writing query results to a file.

The EXIT command returns a value in a batch statement, which can be used to determine the flow of the batch file. If a parameter is input into a batch file, it can be used as part of the query string. The following statement can be used in a batch file, which takes a table name as a parameter and prints the row count of the table. This time the /E option is used for trusted security:

```
osql /E /S TESTSVR /d Northwind /Q "EXIT(SELECT COUNT(*) FROM %1)"
```

The SQL statement used on the command line can be fed in from a file. The output can be redirected to an output file. This is demonstrated in the following example:

```
osql /E /S TESTSVR /d Northwind /i input.qry /o output.txt
```

This works well for pulling information out of SQL Server and batch file processing, but OSQL does not have a facility for importing data. The Bulk Copy Program utility, or BCP, has been optimized to import and export data to and from SQL Server. When using this utility, you need to determine where the data is coming from and where it is going. Unless the data exported will be imported to SQL Server, you should store the data in SQLCHAR format with the –c option. Certain date types such as *timestamp* and *money* are stored in an internal format unique to SQL Server, which limits your ability to work with the data in its raw form. The following statement exports the Customers table to a text file in character format:

```
bcp "Northwind.dbo.Customers" out cust.txt -c -Usa –Ppass -S TESTSVR\
INSTANCE1
```

The first parameter indicates the source of the export, which is a table in this case. A query can be used here if queryout is specified as the next parameter rather than out. The contents of the Customers table are exported to the CUST.TXT file. The –c option indicates that the data should be stored in character format. The –U and –P options take the username and the password, respectively. The –S accepts a server name. With SQL Server 2000, you can install multiple instances of SQL Server. This example references one of those instances.

The BCP utility can also accept a format file for fine-tuned control over the export format. If an export format is not specified, the BCP utility will ask you to enter the format of each column. After the last column format is entered, you will be asked if the format should be saved to a file for later use.

The following statement imports the CUST.TXT file into the Customers table. This sample has been included for illustration purposes only. The import will produce an error if you attempt to run it:

```
bcp "Northwind.dbo.Customers" in cust.txt -Usa -Ppass
-STESTSVR\INSTANCE1 –c
```

We have only scratched the surface of the options available with OSQL and BCP. These utilities have many more command switches that are not covered here. For additional documentation, consult Books Online.

ADO, OLE DB, and ODBC

Microsoft introduced Open Database Connectivity (ODBC) with the intention of providing a consistent interface to database data sources, whether the source is an Oracle database or SQL Server 2000 or some other data-base format. As long as the database

provides a driver that is adheres to the ODBC call-level specifications, it can be called using a common set of functions. This works well for traditional databases, but it does not provide access to nontraditional databases such as Microsoft Exchange or the Active Directory.

Object Linking and Embedding Database (OLE DB) fills this gap. Designed with broader goals in mind, it becomes the data access bridge for spreadsheets, file systems, documents, e-mail systems, and other data-bases using ODBC. Data sources that you might not normally think of as data sources become accessible through the same interface used to access traditional databases. Each data source, including normal databases, has an OLE DB provider, which is the layer between the ADO and the data itself. Information within the data source is returned as rowsets, which most developers are accustomed to dealing with when accessing normal databases such as SQL Server. If a vendor wanted to leverage this technology to give access to network resources, for example, it could write an OLE DB provider designed to access those resources.

Microsoft's ADO provides developers with a powerful object model for programmatically accessing, editing, and updating data sources through OLE DB system interfaces (see Figure 13.4). ADO is commonly used to query a relational database, retrieve and display the results in an application, and allow users to make changes and save the data. Since ADO is composed of a set of COM objects, it can be used in any COM-compatible development environment, including Microsoft's development platforms, such as Visual C++, Visual Basic, and Active Server Pages.

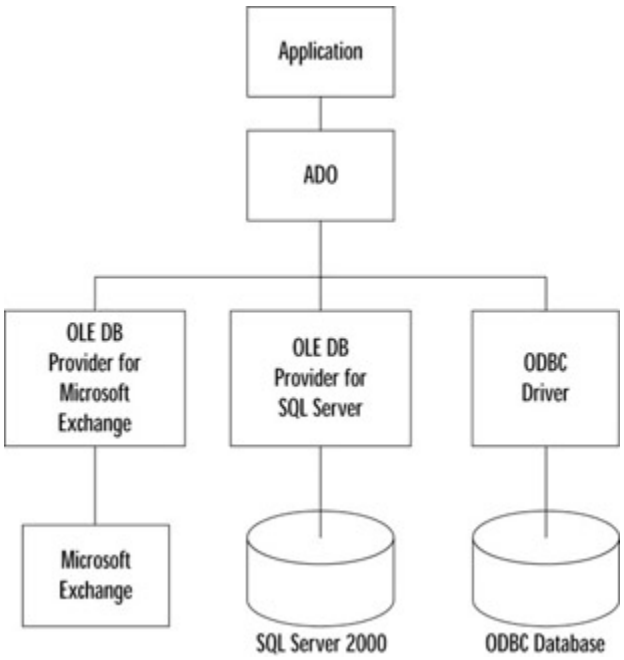


Figure 13.4: ADO, OLE DB, and ODBC.

At the core, ADO contains the Connection, Command, and Recordset objects. The Connection object, as the name implies, establishes connection to the data source. The Command object is used to query stored procedures and user-defined functions. Once a

connection is established and a command has been issued, the Recordset object stores the result set of the command. The Recordset can also be used to query the data source directly. The following Visual Basic sample opens a connection to the Northwind database and reads the Employees table. Before running this sample, make sure to include the Microsoft ADO in your project references:

```
 'Declare local variables
Dim objConnection  As Connection
Dim objRecordset   As Recordset
Dim strConn     As String
Dim strSelect    As String

'Create connection object
Set objConnection = New ADODB.Connection

'Define connection string—update Server, UID and PWD for
'your configuration
strConn = "Provider=SQLOLEDB;Server=TESTSVR\MAINSERVER;" & _
         "UID=sa;PWD=pass;Database=Northwind"

'Open database connection
objConnection.Open strConn

'Create recordset object
Set objRecordset = New Recordset

'Define SQL statement
strSelect = "Select EmployeeID, FirstName, LastName From Employees"

'Open recordset
objRecordset.Open sSelect, _
                  objConnection, _
                  adOpenForwardOnly, _
                  adLockReadOnly




Do Until objRecordset.EOF
    Debug.Print "EmployeeID:" & objRecordset.Fields("EmployeeID")
    Debug.Print "FirstName:" & objRecordset.Fields("FirstName")
    Debug.Print "LastName:" & objRecordset.Fields("LastName")
    objRecordset.MoveNext
Loop

'Close open objects
objRecordset.Close
objConnection.Close

'Clean up object variables
Set objRecordset = Nothing
Set objConnection = Nothing
```

The SQLOLEDB provider is indicated in our connection string. This tells ADO which OLE DB provider to use. Each provider has certain requirements for the connection string, so be sure to consult the documentation associated with the given provider. In this case, the SQLOLEDB provider, which connects to SQL Server, accepts a username and password along with the database and server. If a database is not specified, the user's default database is used. You can also connect using trusted security with the following connection string:

```
strConn = "Provider=SQLOLEDB;Data Source=PLUTO\MAINSERVER;" & _
         "Integrated Security=SSPI;Initial Catalog=Northwind"
```

The Recordset object can use the connection string value directly in the call in place of the Connection object. The Connection object is not required, but it is recommended. You can establish the connection once and reuse it in this particular routine if you need to make additional database calls. However, it should not persist throughout the lifetime of the application, because it uses database and network resources. Microsoft Transaction Server, COM+, and Internet Information Server use connection pooling to quickly establish and share connections between applications. An idle connection serves no purpose in the application and makes it unavailable to other applications in need of a connection.

The Command object takes more code to implement, but it is more efficient than using the Recordset object exclusively, especially when calling stored procedures. If the Command object is used to execute a SQL statement, the query will be prepared and available for reuse. You can think of it as a temporary stored procedure. This is particularly useful when you are executing the same operations repeatedly. The following sample shows one way to call a user-defined function that returns a table:

```
Set objRecordset = New Recordset
Set objCommand = New Command

Set objCommand.ActiveConnection = objConnection
strSelect = "SELECT * FROM ::fn_CustomerSupplierByCity_sb('London')"

objCommand.CommandText = strSelect
objCommand.CommandType = adCmdText

objRecordset.Open objCommand
```

If a stored procedure or user-defined function returns a value other than a rowset, the only way to read it will be through a command object. Recordset objects are unable to access stored procedure return values. This method is a bit more complex. The stored procedure parameters are defined in the Parameters collection of the Command object rather than an SQL statement. The Parameters collection contains Parameter objects indicating the name, data type, length, value, and direction of each parameter. In this case, we need to define a parameter to handle the return value. It needs to be defined as the first parameter in the collection.

In this example, we don't call a stored procedure. Instead, we call a user-defined function. Conveniently, the calling convention for user-defined functions identical to the convention for calling stored procedures. The CommandType property of the command object indicates whether the CommandText is a table, a stored procedure, or an SQL statement. CommandType must be set to adCmdStoredProc, whether you are calling a user-defined function or a stored procedure. The following sample calls the fn_EmployeeOrderCount user-defined function.

```
Dim objConnection     As Connection
Dim objRecordset      As Recordset
Dim objCommand        As Command
Dim strConn           As String

Set objConnection = New ADODB.Connection

strConn = "Provider=SQLOLEDB;Data Source=TESTSVR\MAINSERVER;" & _
         "Integrated Security=SSPI;Initial Catalog=Northwind"

objConnection.Open strConn




Set objCommand = New Command
Set objCommand.ActiveConnection = objConnection

objCommand.Parameters.Append _
    objCommand.CreateParameter("RETURN", adInteger, _
 adParamReturnValue)

objCommand.Parameters.Append _
    oCommand.CreateParameter("FirstName", adVarChar, _
 adParamInput, 10, "Robert")

objCommand.Parameters.Append _
    oCommand.CreateParameter("LastName", adVarChar, _
 adParamInput, 20, "King")

objCommand.CommandText = "fn_EmployeeOrderCount"
objCommand.CommandType = adCmdStoredProc
objCommand.Execute

Debug.Print objCommand.Parameters("RETURN")
```

Warning Beware the *sql_variant* data type when accessing it through different versions of ADO. In the native SQL Server 7.0 version of the native OLE DB provider or the SQL Server ODBC provider, the *sql_variant* is returned as an *nvarchar(4000)* data type. The SQL Server 2000 OLE DB provider returns a field of type *adVariant;* the ODBC provider returns the column as an *adVarBinary* data type when ADO 2.6 is used. The information to remember from this warning is that the *sql_variant* behaves differently when accessed though different OLE DB providers.

Programming Administrative Tasks

Managing databases across an enterprise can be an arduous task. SQL Server includes the SQL Distributed Management Framework (SQL-DMF) to help ease the burden of administration. This framework includes the Data Management Objects (SQL-DMO), Namespaces (SQL-NS), and the Analysis Services. As the name of the framework implies, it is designed to handle a distributed network of SQL Servers.

Distributed Management Objects

The Distributed Management Objects allow you to programmatically control database administration locally and across a network. Since the introduction of SQL-DMO with SQL Server 6.0, it has grown to include over 60 objects and over 1,000 properties and methods. You can administer everything from registering servers to scheduling jobs and managing user access rights. To demonstrate the capabilities of SQL-DMO, Enterprise Manager for SQL Server 2000 uses SQL-DMO to manage SQL Server's entire range of configuration tasks and settings.

SQL-DMO greatly simplifies administrative tasks that would normally be prohibited by time constraints. For example, you could have a list of NT servers but not know which one has SQL Server installed. You could go down the list and try to connect to each one using Enterprise Manager, but that is rather time consuming, depending on how many servers are on the list. This task can be automated using SQL-DMO. Additional information, including a list of databases and their sizes, could be gathered in this way.

At the top level of the object-level hierarchy is the Application object. From here you can get a list of registered servers and SQL Servers in your network (see Figure 13.5). From the SQLServer object, you can drill down into the individual databases on the server. The following example lists the servers registered on the local installation of Enterprise Manager. Make sure that the Microsoft SQLDMO Object Library is included in the references of your Visual Basic project before running this sample:

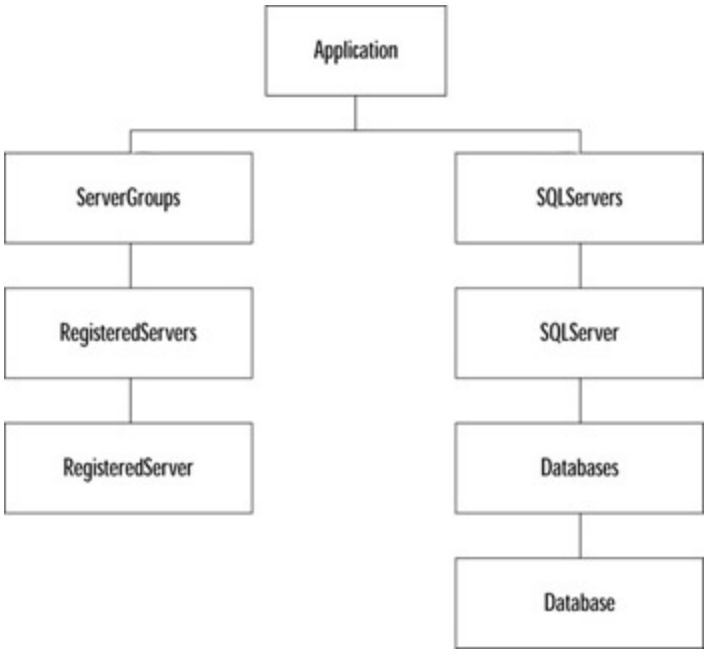


Figure 13.5: The DMO object model.

```
Dim objApplication     As SQLDMO.Application
Dim objSQLGroups       As SQLDMO.ServerGroups
Dim objSQLGroup        As SQLDMO.ServerGroup
Dim objRegServer       As SQLDMO.RegisteredServer

Set objApplication = New SQLDMO.Application

Set objSQLGroups = oApplication.ServerGroups
For Each objSQLGroup In objSQLGroups
    Debug.Print "Registered Group: " & objSQLGroup.Name
    For Each objRegServer In objSQLGroup.RegisteredServers
        Debug.Print " Registered Server: " & objRegServer.Name
    Next
Next
```

Although you can use the Application object as a starting point for obtaining SQL-DMO objects, it is not required. The SQLServer object can be instantiated independently, as can the SQLLogin and SQLUser objects. You can use these objects together to add a user to a database role. The following example adds a user to the db_owner role of all the databases on TESTSVR:

```
Dim objSQLServer     As SQLDMO.SQLServer
Dim objSQLDatabase   As SQLDMO.Database
Dim objSQLLogin      As SQLDMO.Login
Dim objSQLUser       As SQLDMO.User
Dim strUserName      As String
Dim strPassword      As String

Set objSQLServer = New SQLDMO.SQLServer
strUserName = "username"
strPassword = "password"

objSQLServer.Connect "TESTSVR\MAINSERVER", "sa", "sapass"
```

```
Set objSQLLogin = New SQLDMO.Login

objSQLLogin.Name = strUserName



objSQLLogin.Type = SQLDMOLogin_Standard
objSQLLogin.SetPassword "", strPassword

objSQLServer.Logins.Add objSQLLogin

Set objSQLUser = New SQLDMO.User
objSQLUser.Login = strUserName

For Each objSQLDatabase In objSQLServer.Databases
    Set objSQLUser = New SQLDMO.User
    objSQLUser.Login = strLogon

    objSQLDatabase.Users.Add objSQLUser
    objSQLDatabase.DatabaseRoles("db_owner").AddMember strLogon
    Set objSQLUser = Nothing
Next
```

These examples merely scratch the surface of the functionality available with SQL-DMO. You can also administer backup tasks, full-text indexing, transaction logs, and more. For more information about SQL-DMO, see Books Online.

Namespaces

You can integrate elements of the SQL Server Enterprise Manager interface directly into an application with Namespaces. Using SQL-NS, you can display a dialog box to add a user on a remote server. The location of the remote server is indicated in the Initialize method of the SQLNamespace object. The user interface objects are grouped into a hierarchy, which is navigated using identifiers associated with the desired user interface element.

The following sample illustrates the use of SQL-NS to open a dialog box to add a new login. The objects can be found in the Microsoft SQLNamespace Object Library. Make sure you add this library to your references in Visual Basic before running this sample:

```
Dim objNamespaces     As SQLNS.SQLNamespace
Dim varArray(2)       As Variant
Dim objObject         As SQLNS.SQLNamespaceObject
Dim strNamespace      As String

Set objNamespaces = New SQLNamespace
strNamespace = "Server=TESTSVR\MAININSTANCE;UID=sa;PWD=pass;"
```

```
objNamespaces.Initialize "Namespace Object Browser", _
                         SQLNSRootType_Server, _
  CVar(strNamespace)

varArray(0) = objNamespaces.GetRootItem
varArray(1) = objNamespaces.GetFirstChildItem(varArray(0), _
              SQLNSOBJECTTYPE_SECURITY)
varArray(2) = objNamespaces.GetFirstChildItem(varArray(1), _
              SQLNSOBJECTTYPE_LOGINS)

Set objObject = objNamespaces.GetSQLNamespaceObject(varArray(2))
objObject.ExecuteCommandByID SQLNS_CmdID_NEW_LOGIN
```

Analysis Services Programming

Microsoft SQL Server 2000 Analysis Services (formerly OLAP) provide a number of new wizards for simplifying the creation of dimensions and cubes. However, there are some tradeoffs to this ease of use. The user is isolated from some of the more complex features of Analysis Services, such as individual aggregations. This feature and more can be accessed with the Decision Support Objects (DSO), which supply a COM object model usable in Microsoft development platforms such as Visual Basic and Visual C++.

DSO is packaged in a rigid object hierarchy rooted in the Server object. It is the only object that can be instantiated with the New keyword. The rest of the objects must be obtained from a property or method of another object. Collections enforce the relationships in the DSO object model. The OlapCollection object is instantiated by all collections in the DSO object model. It includes the Find method, which is an improvement over standard VB collections and rather handy because the collections do not throw an error if an invalid object is requested. Instead, the object is instantiated as Nothing. This behavior is by design so that the objects can be used in an environment without strong error checking, such as Visual Basic Script:

```
Dim dsoServer       As DSO.Server
Dim dsoDatabase     As DSO.Database
Dim dsoCube         As DSO.Cube
Dim dsoCollection   As DSO.olapCollection
Dim dsoDimensions   As DSO.CubeDimension



Set dsoServer = New DSO.Server
dsoServer.Connect "TESTSVR"
Set dsoDatabase = dsoServer.MDStores("FoodMart 2000")

Set dsoCollection = dsoDatabase.Cubes
If dsoDatabase.Cubes.Find("Budget") = True Then
    Set dsoCube = dsoCollection("Budget")
End If

For Each dsoDimensions In dsoCube.Dimensions
```

```
    Debug.Print " Dimension: " & dsoDimensions.Name
Next
```

The CubeAnalyzer and PartitionAnalyzer objects in DSO correspond to three different wizards used in Analysis Services. The CubeAnalyzer is used independently by the Usage Analysis Wizard, the Partition Analyzer is used independently by the Storage Design Wizard, and both objects are used together for the Usage-Based Optimization Wizard.

The CubeAnalyzer provides access to the query log of a cube. It has only one method, OpenQueryLogRecordset, which returns an ADO 2.6 Recordset object. This is the same output that is produced by the Review Results step of both the Usage Analysis Wizard and the Usage-Based Optimization Wizard. The following sample shows how to use this method:

```
Dim objDSOServer     As DSO.Server
Dim objDSODatabase   As DSO.Database
Dim objDSOCA         As DSO.CubeAnalyzer
Dim objRecordset     As ADODB.Recordset
Dim strSelect        As String

Set objDSOServer = New DSO.Server
objDSOServer.Connect "TESTSVR"
Set objDSODatabase = objDSOServer.MDStores("FoodMart 2000")

Set objDSOCA = objDSODatabase.Cubes("Budget").Analyzer
strSelect = "Select * From QueryLog"
Set objRecordset = objDSOCA.OpenQueryLogRecordset(strSelect)
```

The PartitionAnalyzer object is accessed through the Analyzer property of the DSO Partition object, similar to the way the CubeAnalyzer is accessed through the Analyzer property of the Cube object. It can create aggregations for a partition in two basic modes. First, it can determine if the designed aggregation can satisfy a given query using the AddGoalQueries and PrepareGoalQueries methods. In the second mode, the PartitionAnalyzer uses a mathematical simulation to create goals against which aggregations can be tested.

DTS Programming

The Data Transformation Services (DTS), introduced with SQL Server 7.0, provide a set of tools to easily extract data from various heterogeneous data sources, transform it, and store it in SQL Server or another database altogether. DTS packages can be created and manipulated using a set of DTS COM components.

You can access, create, and run DTS packages using these objects in a Windows development platform that supports COM. The Application object is the root of the DTS object hierarchy (see Figure 13.6). The packages are stored in either SQL Server or the Meta Data Services. The two methods of the Application object, GetPackageSQLServer and GetPackageRepository, access these sources, respectively. The PackageRepository object contains more information about the contained packages, such as the lineage and version; the PackageSQLServer object has a reduced set of options.

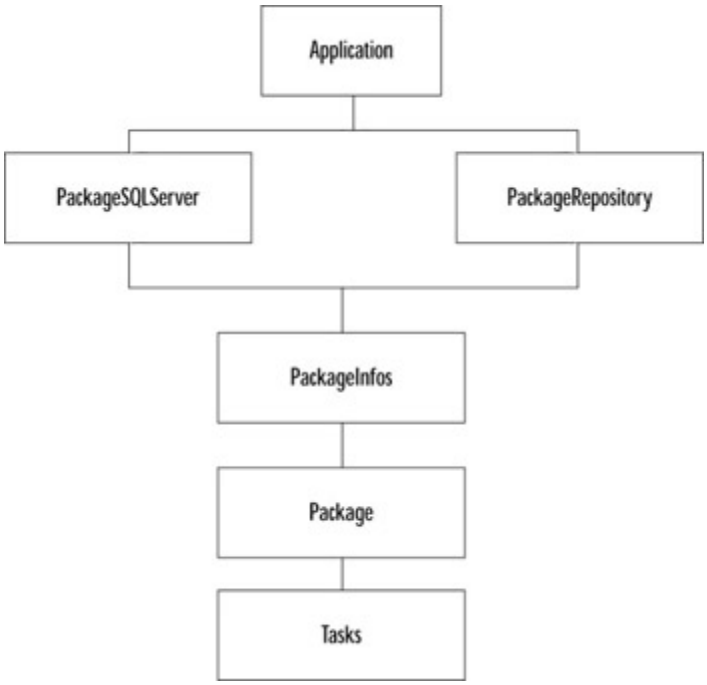


Figure 13.6: The DTS object hierarchy.

Note SQL Server 2000 extends the programming interface to the DTS components with the Parallel Data Pump task to handle rowsets. It can be used only programmatically through the DTS objects. It cannot be used within the graphical DTS package designer.

The following sample lists the tasks and connections in a given DTS package. Ensure that the Microsoft DTS Package Library is selected in the project references prior to executing this code. The LoadFromSQLServer method of the Package object can reference a DTS package using the globally unique identifier (GUID) or the package name. If the version GUID is not specified, the latest package is loaded:

```
Dim objPackage   As DTS.Package
```

```
Dim objTask     As DTS.Task
Dim objConnection  As DTS.Connection

Set objPackage = New DTS.Package

objPackage.LoadFromSQLServer "TESTSVR\MAINSERVER", "sa", "pass", _
                               DTSSQLStgFlag_Default, , , , _
                              "TestPackage"

Debug.Print "Package GUID: " & objPackage.PackageID
Debug.Print "Version GUID: " & objPackage.VersionID

For Each objTask In objPackage.Tasks
    Debug.Print objTask.Name
Next

For Each objConnection In objPackage.Connections
    Debug.Print objConnection.Name
Next

objPackage.Execute

objPackage.UnInitialize

Set objPackage = Nothing
```

Replication Programming

The Replication objects are part of SQL-DMO. They allow you to control SQL Server's data replication functions. You can set up and control publishing, distribution, and subscription roles of SQL Server data replication. Each Replication object contains the Publisher, Subscriber, and Distributor object collections, as shown in Figure 13.7 .

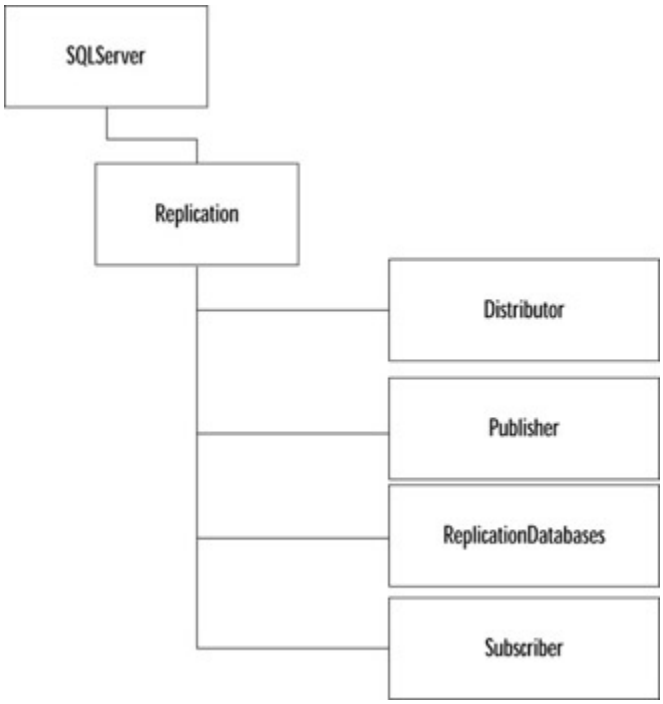


Figure 13.7: The SQL-DMO Replication object model.

Meta Data Services Programming

In December 1998, Microsoft transferred rights to maintain and evolve the Open Information Model to the Meta Data Coalition (MDC), an industry consortium comprising dozens of vendors of enterprise tools, metadata management tools, and data warehousing products. The MDC determines the Open Information Model (OIM) standard, which is a vendor-neutral metadata standard in much the same way OLE DB is a data source-independent access standard. In July 1999, the MDC released version 1.0 of the MDC Open Information Model (OIM) as a standard. (For more information about this standard, see www.mdcinfo.com.)

The OIM is a standard interface through which repository access is done. It defines the structure of the information in the repository in much the same way table schema in a database defines how row data is stored. The Microsoft Repository type library contains the objects used to access the OIM.

The Repository object in this library is the main access point of entry to the repository. The RepositoryObject represents each item in the repository. The following sample opens the SQL Server 2000 repository located in the MSDB database and steps through the interfaces accessible through the root node:

```
Dim objRepository    As RepositoryTypeLib.Repository
Dim objRoot          As RepositoryTypeLib.RepositoryObject
Dim objClassDef      As RepositoryTypeLib.RepositoryObject
Dim objInterface     As RepositoryTypeLib.RepositoryObject
Dim strRepDB         As String

Set objRepository = New RepositoryTypeLib.Repository
strRepDB = "SERVER=TESTSVR\MAINSERVER;DATABASE=msdb",
Set objRoot = objRepository.Open(sRepDB, "sa", "sapass")
Set objClassDef = objRepository.Object(oRoot.Type)



For Each objInterface In objClassDef("IClassDef").Interfaces
    Debug.Print "Interface: " & objInterface.Name
Next
```

Summary

In this chapter, we have explored several new programming enhancements to SQL Server 2000 and reviewed some existing technologies. We covered the expanded range of identity values the new *bigint* data type permits and its related @@ROWCOUNT_BIG and BIG_COUNT functions. You learned about the flexibility of the *sql_variant* data type and the rules surrounding converting to and from this data type, as well as the proper techniques for comparing *sql_variant* variables to variables of other data types. Recall that the *table* data type cannot be used as the data type of a rowset column, but it can be used as a variable in a stored procedure or user-defined function. Because it does not occupy space in the transaction log, it proves an efficient alternative to temporary tables.

The new Query Analyzer has a number of enhancements developers will find useful immediately. Templates boost productivity by providing skeletal scripts for common operations such as creating a table or iterating through a cursor. You saw that the database schema is viewed directly in Query Analyzer. You no longer have to flip between Enterprise Manager or use system stored procedures to find schema information. You debugged a stored procedure directly in Query Analyzer.

With prior versions of SQL Server, data in related tables had to be maintained using either stored procedures or triggers. A DELETE trigger on the Orders table could delete the related rows in the Order Detail table when an Order is deleted. Now, with SQL Server 2000, you can use cascading updates and deletes to simplify relational maintenance.

We covered the three types of user-defined functions—scalar, table, and inline—and how they can be a powerful alternative to stored procedures. The scalar user-defined function provides a convenient means of returning a single value. For example, you could use it to return the number of days remaining in the year, given the date. The table and inline user-defined functions both return table data types. The difference between the two lies in the fact that the table user-defined function can contain multiple internal statements; whereas the inline function simply defines a single SELECT statement.

We have seen that views have been enhanced with support for indexes and INSTEAD OF triggers. Views are now more than a handy way to hide data or consolidate tables. Although numerous restrictions govern which views are eligible for an index, the enhancements make it worth adhering to those restrictions. Indexes placed on columns used as the basis for a join can more than double the speed of a query. We also used the INSTEAD OF trigger to update multiple tables underlying a view.

You received an introduction to Meta Data Services, an evolving and increasingly popular technology. Perhaps you will not see much of it in the workplace this year, but it is gaining ground and building momentum, so you should at least be aware of and acquainted with it.

We reviewed some existing technologies that have been carried over from prior versions of SQL Server. We reviewed the ISQL command-line utility for executing SQL scripts. The new OSQL command is very similar to ISQL but differs in that it uses ODBC rather than the DB-Library to communicate with SQL Server. We also touched on the BCP command for doing bulk inserts. Although it is an old command, it is still faster than some of the newer importing options in certain cases. If you are importing a single comma-delimited text file, a simple batch file using BCP would consume much less overhead than creating a DTS package to perform the same task.

Entire books have been written about Transact-SQL and ADO. This chapter simply provides an introduction to these technologies. There is much more to learn; you are certainly encouraged to continue exploring these powerful and popular technologies. If you are new to either T-SQL or ADO, this chapter provided a foundation on which to build. We covered the basics of the SQL statements and some of the differences between SQL 92 and T-SQL. We also flexed some of the muscle of ADO and saw how to access SQL Server 2000 using OLE DB.

We covered some of the breadth of the SQL Data Management Foundation, which includes the Data Management Objects (SQL-DMO), Namespaces (SQL-NS), Decision Support Objects (DSO), and Data Trans-formation Package components (DTS). With these technologies, database administrators and programmers alike gain definite and fine-tuned control over maintenance and administrative tasks.

Although some of these technologies are not new, they could be new to you. Most developers have little need to learn SQL-DMO or SQL-NS unless it is specifically required for a project. Every developer adds to a toolbox during his or her career. If a tool is not necessary at the moment, it doesn’t mean it’s useless. More options allow you to make better-informed decisions.

FAQs

Q. My SQL statement works correctly until I place it in a user-defined function. Why?

A. You have probably included a statement that has side effects. User-defined functions are intended to return information, not change data outside the scope of execution. Therefore, external tables cannot be modified, e-mail cannot be sent, and can database objects cannot be created.

Q. In SQL Server 7.0, if you needed to load a temporary table with the results of a stored procedure, you needed a statement that looked something like INSERT INTO #TableName EXEC sp_proc. Can you load the result of a stored procedure into a *table* data type instead?

A. No. The table data cannot use the EXECUTE function as the source of an insert.

Q. Can columns not directly referenced in a schema bound view be altered?

A. Yes. If a five-column table is schema bound to a view that references only the first two columns, the latter three can be altered.

Q. Views cannot be schema bound to other views. Can I create a view WITH SCHEMABINDING if it is referencing another schema bound view?

A. Yes. If a view is created WITH SCHEMABINDING, it can be referenced by other views created WITH SCHEMABINDING. The same restrictions apply to views as they do to tables. If a view is schema bound to another view, it cannot be changed until the referencing view removes the binding.

Q. I can define system-level stored procedures, which can be used in all databases on the server. Can I define system-level, user-defined functions?

A. Yes. The function must be created in the master database with the fn_ prefix. Ownership of the function must be granted to system_function_schema. The following statements demonstrate the creation and use of a system-level, user-defined function. Note that the function referenced in the SELECT statement is not qualified with the function owner name:

```
USE master
GO
CREATE FUNCTION fn_DayOfYear(@date datetime)
RETURNS int
AS
BEGIN
  RETURN CAST(DATEPART(dy,@date) as int)
END
GO
EXEC sp_changeobjectowner 'fn_DayOfYear',
'system_function_schema'
            GO
            USE Northwind
            GO
            select fn_dayOfYear(GETDATE())
```

Chapter 14: Performance-Tuning Tools and Techniques

Introduction

The success of nearly every application is determined by its reliability and performance. Squeezing every millisecond out of a solution is an ongoing task for most application designers and administrators. With SQL Server 2000, Microsoft continues to add performance enhancements to SQL Server with features such as auto-tuning capabilities that allow SQL Server to monitor activity and perform self-tuning tasks. Smaller organizations without dedicated administrators can enjoy greater out-of-the-box performance than with previous versions of SQL Server. For advanced tuning, SQL Server has enhanced the SQL Profiler utility to include additional trace methods and events for recording and analyzing database activity. Tools such as the Index Tuning Wizard can analyze and alter indexes for optimal query performance.

A significant enhancement in SQL Server 2000 is its support for federated database servers. New to SQL Server 2000, federated servers allow for data partitioning across multiple independent servers, so processing is distributed among the federated members. This scale-out technique was used to achieve new performance records for SQL Server and record top scores in Transaction Processing Council (TPC, www.tpc.org) testing, the leading indicator of database performance.

Optimizing SQL Server performance is not limited to SQL Server alone. As the first database system “designed” for Windows 2000, SQL Server’s optimal performance means tuning Windows 2000 to support the tasks of SQL Server. With every application, performance starts with efficient logic, and SQL Query Analyzer includes graphical execution plan analysis for creating optimal T-SQL code.

This chapter reviews the performance enhancements in SQL Server 2000 and techniques for monitoring and increasing performance in your SQL applications.

Partitioning Data and Federated Database Servers

SQL Server has always been the easiest database to work with and a low-cost solution affordable to most organizations. With the delivery of SQL Server 2000, it is now the fastest database in the world, as determined by the TPC. The TPC validated that SQL Server 2000 on Compaq hardware running Windows 2000 achieved a transactions-per-minute (tpmC) score of 505,302. In order to achieve such a score, a database must scale to great heights. There are two fundamental approaches to server scalability:

- Scaling up
- Scaling out

Scaling up involves making your database server a more powerful machine by adding more and faster processors, memory, and disks. This has been the traditional way to increase the performance of SQL Servers, but often a limit is encountered when the physical capabilities of the hardware are reached. In previous releases, scaling up was the primary scalability method of SQL Server. In addition, SQL Server 2000 supports Microsoft's latest scalability strategy, called *scaling out.* Scaling out allows you to increase the performance of your databases by allowing multiple SQL Servers to share the workload, called a *federation* . A federation is made up of *federated servers,* or servers that are administered independently but that work together to share database processing activities.

The secret to creating a SQL Server federation is horizontally partitioning the data across the servers in the federation. Each member of the federation has some of the data locally and a view of all the data from all the machines. This approach is called *share nothing* , because none of the servers shares common resources such as disk arrays. With this strategy, you don't build a bigger server to handle an increased load; you add extra servers to handle the burden.

Note Using federated servers is not an availability solution. If a server fails, its participation and data are lost. Federated servers are a scalability solution designed to service enormous workloads.

Overview

The key to an effective federation in SQL Server 2000 is a distributed partitioned view. In order to create a distributed view, you need to create a table on each machine that will hold part of the information. These data structures are called *members* . If we wanted to create a distributed partitioned view of the Orders table in the Northwind database, we could do so as illustrated in Figure 14.1 . Order_1, Order_2, and Order_3 are the member tables of the view Orders. They are made to appear as one table by joining them all together in a view using the UNION operator with the ALL parameter. The view described in Figure 14.1 would be created with a statement like this:

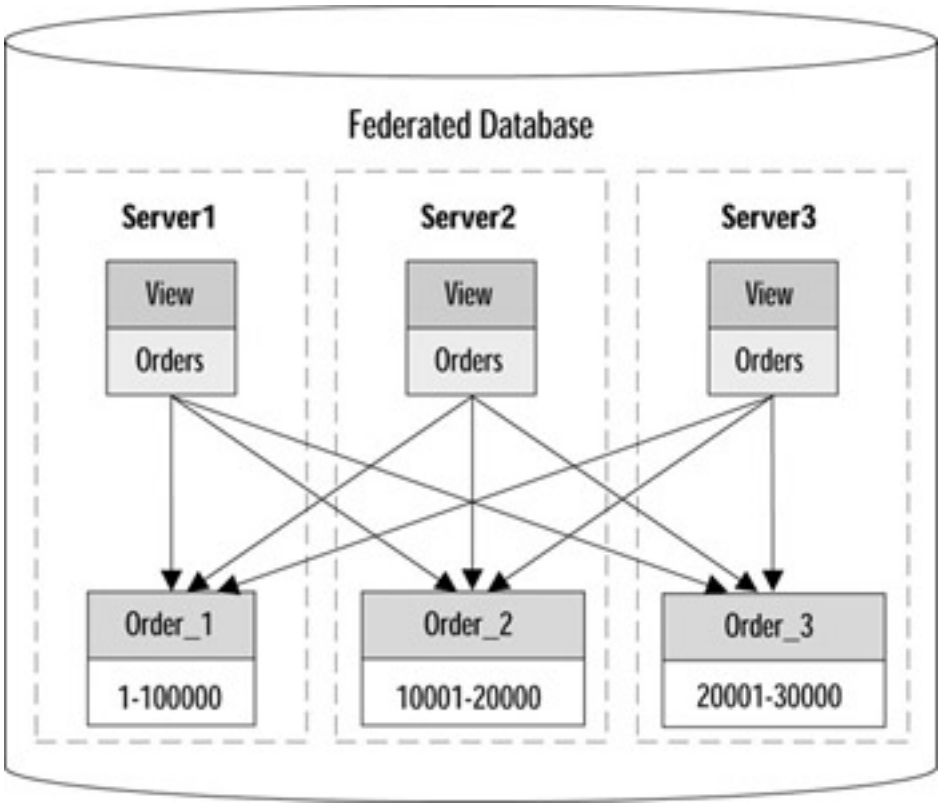


Figure 14.1: An example of a federated database.

```
CREATE VIEW     Orders AS
                SELECT *
                FROM Server1.Northwind.dbo.Order_1
        UNION ALL
                SELECT *
                FROM Server2.Northwind.dbo.Order_2
        UNION ALL
                SELECT *
                FROM Server3.Northwind.dbo.Order_3
```

As you can see, each member table is combined using the UNION ALL statement to form a view that that is distributed across multiple servers. If you tried to insert a record into the view, in which member table would the record reside? There is no WHERE clause to direct records to each table, so how does SQL Server find a given record? Does it have to search each table? The answer to the last question is no; each member table must have a partition column with a CHECK constraint on it that sets a valid range of values for this table. SQL Server uses the ranges of the CHECK constraints to determine the location of the server on which the data resides.

There is another problem, though. Looking at Figure 14.1 , you can see that the Order table in the Northwind database is horizontally partitioned on the key OrderID. OrderID is generated from an IDENTITY. If you include an IDENTITY field in your distributed view, you will not be able to add new records. If you want to distribute a table like Orders across a federation, you will have to remove the identity columns in them or they will be

read only. You can replace the values generated in the IDENTITY column with an algorithm in your middleware to generate similar unique values.

In order to achieve the greatest benefit from organizing your data into distributed partitioned views, you should organize your structures so that you reduce the amount of interserver communication. For instance, suppose that your order information is kept in two tables: Orders and Order Details. If OrderID relates Order Details to Order by the foreign key OrderID and you partition Order, you should also partition OrderDetail by OrderID. This will keep the related data on the same server for fast retrieval by ensuring that as much of a query as possible can be executed on the local server, as opposed to a remote server. Table 14.1 provides some guidelines on partitioning.

Table 14.1: Guidelines for Partitioning

Partition Across Servers	**Clone to Each Server**	**Isolate to a Single Server**
If more than 5 percent of all statements referencing a table involve data modification and the table can be effectively partitioned along the desired dimension	If less than 5 percent of all statements referencing a table involve data modification	If more than 5 percent of all statements referencing a table involve data modification and the table cannot be effectively partitioned along the dimension you have chosen

Some data cannot be distributed effectively and will be needed by all servers in the federation. For instance, Orders might be related to a table called OrderStatus. OrderStatus stores the type of order in Orders, and most queries against Orders will require a join to OrderStatus. If OrderStatus were distributed across the federation, most queries would require executing part of those queries remotely. If you place OrderStatus on a single server, that server would have to execute all its queries locally, but all the other servers would suffer from network lag. Instead of partitioning OrderStatus or locating it on a single node, you should clone OrderStatus across all the members of the federation. This will provide local access for all the servers.

Note Using the ALL parameter with UNION ensures that all rows, including duplicates, are included. With the ALL parameter, duplicate rows would not show up in the results. By including all rows, SQL Server does not have to sort the results, which would require a great deal more overhead.

The price you will have to pay for this local access across the servers is that you must define a way to keep all the data synchronized. If a high degree of transactional integrity is important for the partitioned tables, you can create triggers that propagate the changes to each server. If integrity requirements are not as stringent, you can use SQL Server replication to keep the tables synchronized.

Basic steps create a distributed partitioned view:

1. Design and create the member tables on each server. For each server, you must create linked-server definitions with the connection information for the other servers in the federation and set the lazy schema validation option for each linked server definition used in distributed partitioned views.
2. Create a distributed partitioned view on each server in order to make the location of the data transparent.

Designing Your Tables

Not all tables can be part of a distributed partitioned view; there are several restrictions on the tables and columns that can participate. The member tables must all be configured alike. They must have the same ANSI padding setting, and each must have a primary key. In addition, each primary key in a member table must have the same number of columns. Although the tables can contain computed columns, there must be no index on any of the computed columns in any of the member tables.

There are several restrictions involving the columns from the member tables. Unlike ordinary views, in a distributed partitioned view, it is not enough to have the columns in the member tables be implicitly convertible data types. They all must be of the *exact* same type. It is not enough for two unioned columns to be numbers; they must have the same precision and scale. It is not enough for two unioned columns to be strings; they must be the same type and length, and they must have the same collation. For instance, if one of your tables had a numeric (5,3), it could not match up with a column that was numeric (5,2). Another thing to note is that *smallmoney* or *smalldatetime* data types are mapped as *money* and *datetime* data types, respectively, on the remote servers. Due to this behavior, you should not use *smallmoney* or *smalldatetime* data types in tables that will be used for distributed partitioned views.

While designing your tables, you must designate one column as the *partition column* . The partition column is the column that identifies the range of data available in that specific table for the view. Due to its role in partitioning the data, the partitioning column carries additional restrictions, the most critical of which is that the range of valid values for the partitioning column are imposed by an enabled CHECK constraint. This information is used by the query processor to determine which member tables contain which data. This constraint must be the only one on this column. If there is more than one, all of them will be ignored. In addition to the CHECK constraint, the partition column cannot contain NULL values, must be part of the key, and cannot be a computed value.

If you construct your tables with these restrictions, they will be able to participate in a distributed partitioned view. However, this ability will ensure only that you will be able to run SELECT statements against the view. To support UPDATE, DELETE, and INSERT statements, there are several additional restrictions. None of the columns can be the *timestamp* data type or have a column that generates its value from an IDENTITY. In addition, the columns in your primary key cannot be of the *text, image,* or *ntext* data types if you want to be able to update the fields of the primary key through the view. If they are

of these types, you will be able to modify the view as a whole, but you will not be able to modify any of the columns that make up the primary key. You can ignore these constraints by writing INSTEAD OF triggers. Using INSTEAD OF triggers, you can insert directly into the correct tables, bypassing the view, but they will not always choose the most efficient query planes.

Checklist for tables to participate in a partitioned view:

- All member tables must have the same ANSI padding.
- All member tables must have the same primary key.
- The member tables cannot be an index on any of the computed columns.
- Columns must be of the *exact* same data type, precision, scale, and length.
- Member tables should not use *smallmoney* or *smalldatetime* data types.
- The partition column should have only one constraint.
- The partition column cannot contain NULL values.
- The partition column must be part of the key and cannot be a computed value.

In addition, to allow the view to be updatable:

- None of the columns can be the *timestamp* data type.
- None of the columns should have its value generated from an IDENTITY.
- The columns in your primary key should not be of the *text* , *image* , or *ntext* data types.

Configuring the Servers

As illustrated in Figure 14.2 , each server must be configured to communicate with the other servers in the federation. To do this, provide a linked server definition on each server with information about how to use OLE DB to “talk” to the other servers. For instance, Server1 must have definitions for Server2 and Server3. Likewise, Server2 must have definitions for Server1 and Server3, and Server3 must have definitions for Server2 and Server3.

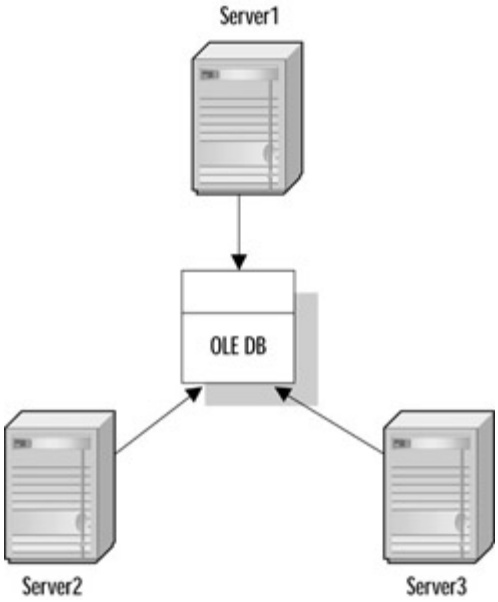


Figure 14.2: Communication among linked servers.

As with most tasks that you need to accomplish in SQL Server, you can either use Enterprise Manager or Transact-SQL to link the servers together. You can use the sp_addlinkedserver system-stored procedure to add linked server definitions on Server1 for Server2 and Server3 as described here.

If you specify SQL Server as the product, the additional provider properties are not valid. Use the following example to establish a linked server to the Northwind database on SRV3 or a root linked server to SRV3. Update the server names according to your examples:

```
EXEC sp_addlinkedserver @server='SRV3_NWND', @srvproduct='',
@provider='SQLOLEDB', @datasrc='SRV3', @catalog='northwind'
```

Or:

```
EXEC sp_addlinkedserver @server='SRV3', @srvproduct='SQL Server'
```

Using Enterprise Manager, you can follow these steps to add a linked server:

1. Right-click on Linked Servers under the Security folder for the server to which you want to add the linked server.
2. On the right-click context menu, click New Linked Server… to display the Linked Server Properties—New Linked Server dialog box.
3. Enter a **Linked Server name** in the Linked Server text box.
4. Select the server type. You can select either SQL Server or Other Data Source. If you select Other Data Source, you must enter the information that OLE DB needs to communicate with the remote database. Because you can link to a variety of databases that have OLE DB providers, the required information changes with respect to the database to which you are linking. However, it is some combination

of Product Name, Data Source, Provider Name, Location, and Catalog. As you click your cursor in each required text box for that OLE DB provider, a frame below the box describes the information that is required to use that data source.

After you have defined each of the remote servers, you must alter the way that the member servers interact with each other. Specifically, you want to set the *lazy schema validation* option for each of the linked server definitions on every member server. The only way that this can be done is with the sp_serveroption system-stored procedure. This is a requirement because it prevents the query processors in the federation from requesting metadata from any of the remote member tables until it is needed, which greatly reduces communication. Before you try to use sp_serveroption, you must make sure that you are a member of the sysadmin or setupadmin fixed server roles. You could correctly configure the linked servers definitions for Server2 and Server3 on Server1 by running the following Transact SQL statements:

```
Exec sp_serveroption 'Server2', 'lazy schema validation', 'True'
Exec sp_serveroption 'Server3', 'lazy schema validation', 'True'
```

After you have done this, you will have a read-only distributed partitioned view, which is valuable for reporting, analysis, or data delivery applications, but the underlying tables cannot be updated using the view. If you want to be able to execute INSERT, UPDATE, and DELETE statements against the view, you must set the XACT_ABORT SET option to ON. Setting the XACT_ABORT SET option to ON will force transactions to roll back if you encounter a runtime error. You can set this option by running this statement in the query analyzer:

```
SET XACT_ABORT ON
```

Note Updateable distributed partitioned views require SQL Server 2000 Enterprise or Developer Edition. Enhancements to SQL Server 2000's query optimizer and updateable partitioned view capabilities offer full support for scale-out capabilities. Partitioned views are not limited to Enterprise and Developer Edition. SQL Server 2000 Standard Edition supports read-only partitioned views that are useful in many reporting and analysis applications.

Creating the View

A *view* is a stored SELECT query that appears and can be referenced as a table. A *distributed view* is a view that spans multiple servers, and like the tables that make up a distributed view, there are restrictions on the view itself. The view must be created using UNION ALL to unite the member tables, and each member table can be referenced only once in the SELECT statement that creates the view. Each column of each member table must be included in the view once and only once. In addition, each column must be in the same ordinal position in each of the SELECT statements, and the view cannot contain columns with derived values and aggregate functions.

Optimizing Query Performance

All queries are not created equal. Some perform better than others because they are able to take advantage of existing resources and optimal query techniques.

The process of optimizing query performance has many facets. One popular technique for improving query performance is the use of indexes to speed up the retrieval of data in large, diverse tables. Indexes are not the only solution to improving query performance, however. Determining which queries are performing poorly and analyzing them is a task with which all SQL developers should be familiar. SQL Server 2000 provides utilities for performing both of these tasks. SQL Profiler allows you to monitor and record query activity, including start and finishing times and SQL syntax for your queries. You can even combine your SQL Profiler trace file with SQL Server 2000's Index Tuning Wizard to automatically analyze your database and allow SQL Server to make recommendations on indexes. After you have identified poorly performing queries, you can use the enhanced Query Analyzer to manually analyze performance statistics and execution plans so that you can fine-tune your syntax and database objects.

Understanding Indexes

Without a good understanding of indexes, it would be very difficult to create a well-functioning relational database solution. Indexes exist to help data retrieval performance. The easiest way to understand indexes is to imagine going to the library and trying to find all the books on SQL Server so that you can study for a test. Suppose this library has no card catalog and does not use the Dewey decimal system. You would have to search the entire library and look through each book—a huge amount of searching to find just the handful of books that you need. However, in a library that implements the Dewey decimal system and has card catalogs, you will be able to quickly find the books you need.

A table in SQL Server without any indexes is analogous to a library that does not use the Dewey decimal system or card catalogs. Card catalogs are the indexes to the books in the library. They organize the books by author, subject, or title, making it unnecessary for you to scour the library for days on end to find what you are looking for. Likewise, a SQL Server index can often help you find exactly what you need in a fraction of the time that it would take to read an entire table. Indexes do this by keeping track of the data in particular locations, dramatically decreasing the amount of reading that SQL Server must do to find data. This takes some extra storage space and some extra writing to the disk to maintain, but since the vast majority of operations in the database are read operations, this proves to be an excellent trade-off.

Indexes are one of the most important parts of the physical database architecture; if carefully thought out and tweaked on occasion, they can maximize the performance of your database. However, if poorly thought out or neglected, they can hobble the performance of your systems. A *table scan* —the process of looking at each and every row in a table to find whatever you are looking for—happens whenever a query cannot

find a useful index. Without an index, SQL Server would have to look through each row because it would know if a value exists only if you look at each and every possible value. This might work on a table consisting of 16 rows, but what about a table with 10 million rows? This is not only true of SELECT queries but also DELETE and UPDATE queries because they must find the data before they can remove it or modify it.

So why not index everything? There are several reasons. For one, each index adds overhead. Every index on a table must be updated every time an INSERT or DELETE is committed, and some of them might have to be maintained after UPDATEs are performed. In addition, indexes consume disk space. Finally, if you place indexes on columns with very few unique values, using the index could be slower than performing a table scan of the same table.

Types of Indexes

There are three important equivalence classes of indexes that you can use:

- Clustered
- Nonclustered
- Covered

Clustered Indexes

Recall the library analogy we just discussed. It might seem that the Dewey decimal system is a little bit different from the card catalogs. The Dewey decimal system is the system for the physical layout of the books in a library; it controls the physical order of the books. When you find a book using the Dewey decimal system, it is right there in front of you. However, when you find a book with a card catalog, you find a Dewey decimal number, then have to use that number to find the actual book.

The *clustered index* of a table is comparable to the Dewey decimal system. It is the most important index you place on a table, because it specifies the physical order of the data in the table. In Figure 14.3 , the indexes are represented by the gray columns. As you can see, the clustered index, column A, controls the physical layout of all the data in the table, including the columns that are not part of the index. As a consequence, there can be only one clustered index per table because the data in a table can be physically stored only one way. A clustered index is very efficient for columns in which you search for continuous ranges of values, because all the values are in sequential order. The clustered index on the Orders table in the Northwind database is indexed on OrderID. A query such as this would be an ideal opportunity to take advantage of that clustered index:

```
SELECT *
FROM  Orders
WHERE OrderID >= 10281
AND  OrderID < 10755
```

In addition to being ideal for queries that return a range of values, clustered indexes are also very useful for queries that sort their results or provide grouped results:

```
SELECT  OrderDate, OrderID
FROM   Orders
WHERE   OrderID >= 10755
ORDER BY OrderDate
SELECT  OrderID, COUNT(OrderID) AS TotalSales


FROM   Orders
GROUP BY OrderID
```

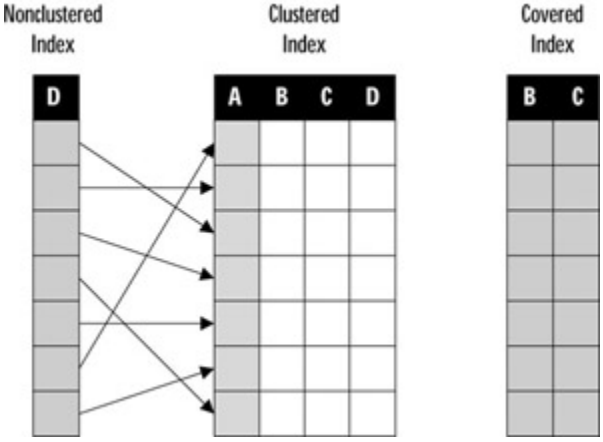


Figure 14.3: Classes of indexes.

This is because the query engine will not have to take the time to sort the values after it finds them, because the clustered index has them already sorted. As a general rule, clustered indexes should be tightly defined and cover as few columns as possible, because all nonclustered indexes contain the clustered indexes within them. If you make the clustered index five columns wide, each nonclustered index will contain its own fields plus the five columns from the clustered index. In addition to requiring more storage, having numerous columns within a clustered index increases the likelihood that one of those fields will change. If a field in a clustered index changes, the whole row will need to be moved to a different location—which will increase the number of DELETEs and UPDATEs that your system will perform.

Another thing to consider when choosing your clustered index is to try to *avoid sequentially increasing fields* as the clustered index. For example, if you make an IDENTITY field, which is sequentially increases your clustered index, your table will suffer from frequent page splits under heavy insert activity. In a monotonically increasing field, the values always get higher. If you organize your data by that field, the highest value is always going to be the last value on the last page. After a few inserts, the 8K data page is filled and a page split occurs causing a new page to be created. The approximately half of the full page's data is moved to the new data page. This is called a *page split.* Page splits require time and will slow down INSERT operations into the table.

Now, with this information, you can see that OrderID is a poor choice for the clustered index on Orders.

Another thing you can do to minimize the number of page splits is to set the correct fill factor for your clustered index. The *fill factor* is a percentage from 1 to 100 that determines how full the 8k data pages should be when the index is created. 0 is also a valid value, but it is not interpreted as a percentage. It is treated similarly to a value of 100 with the exception that it will leave some room on each page. A value of 50 would instruct that the index be created with approximately half of each data page full of data and the other half empty. This can reduce page splits by having room on each page, but a fill factor of 50 would take almost twice as much space as a fill factor of 100, making reads much slower, and reads always outnumber writes due to the fact that the computer must find the location in the table in order to write the information. Because of the adverse effect on reading data, it is recommended that you leave the fill factor at its default value, 0, which will make the pages completely full.

It is often suggested that clustered indexes should be created on a table column that contains a limited set of distinct values, such as the state abbreviation. This advice works very well if you happen to analyze your data by state abbreviation and if you have a good distribution of states in your data. But if 90 percent of state abbreviations you capture in your sales log are "CA," this index is not going to help you very much.

Tip SQL Server 2000 has added a new ability to index views in addition to tables. This ability can allow you to have precomputed aggregate values and to simulate additional clustered indexes.

Nonclustered Indexes

If the Dewey decimal system is analogous to clustered indexes, card catalogs are similar to *nonclustered indexes* . Just as there can be many card catalogs (each listing the books from a different perspective), you can have many indexes on a table, organizing the data in different ways.

A nonclustered index is any index that does not determine the physical order of the data. Like the card catalog, nonclustered indexes are usually tailored to a specific purpose. If you like an author and want to read more of his works, you would use the author card catalog to find them, but if you knew the title of the book you wanted, you would use the title card catalog. There would be cards for both books in each catalog, but you use them in different circumstances.

Whereas clustered indexes excel at returning ranges of data, nonclustered indexes shine in finding specific pieces of data. They typically can isolate individual pieces of data very quickly, but they do not control the layout of the data. When you find the right book using the card catalog, the book is not in front of you, but you know where it is in the library. The same is true of a nonclustered index; it tells you where to go to find the information, whereas with the clustered index, the information is presented to you.

With the nonclustered index, you have the extra step of going out and getting the data, making it slower than the clustered index. Since the data is laid out in the order of the clustered index, once you find the values you are looking for in the nonclustered index, you will then have to go to find the correct place in the table using the clustered index. As you saw in Figure 14.3 , once you have found the correct value in the nonclustered index, you must then look up the rest of the information in the table using a row locator. If you had wanted all the books written by Isaac Asimov, for example, you could quickly find them in the card catalog, organized by author. However, since the Dewey decimal system is organized by subject, and Asimov wrote on a multitude of subjects, you would be collecting books from all over the library. You could find each individual one quickly, but since they are in different locations, you would have to make many trips.

Covering Indexes

A *covered index* is a nonclustered index that contains all the columns in a query. It is analogous to using the card catalog but not needing the book because the card has all the information that you need. Covered indexes must have all the fields in the WHERE clause plus all the fields in the SELECT clause. If a great deal of our use of the Orders table in the Northwind database had to deal with retrieving the CustomerID by OrderDate, for example, we could build a clustered index like this:

```
CREATE INDEX idxCovered ON Orders(OrderDate,CustomerID)
```

To help with queries like this:

```
SELECT CustomerID
FROM  Orders
WHERE  OrderDate = '9/20/96'
```

A covered index dramatically reduces the number of reads needed, because the query engine never has to follow the pointers to the data pages, since all the information is contained within the index. In addition, because all the data are contained within the index, it behaves like a clustered index in the sense that all the data are laid out sequentially, like a clustered index. But the index performs better than a clustered index because it is smaller, since it has fewer columns.

General index rules:

- Choose the clustered index wisely. There can be only one, and it controls the physical layout of the table.
- Be very careful in changing the clustered index fill factor away from the default of 0.
- Avoid monotonically increasing fields as the clustered indexes.
- Single-column integer values make good clustered indexes.
- A clustered index should be on a column that is not unique and that has a well-distributed range of values.
- Clustered indexes are good for queries that retrieve ranges.

- Clustered indexes are good for queries that use aggregates.
- You can have multiple nonclustered indexes.
- Nonclustered indexes are best on columns with a high percentage of unique values.
- A covered index is the fastest class of index.

Optimizing Database Performance with SQL Profiler

SQL Profiler, as shown in Figure 14.4 , is a powerful tracing tool that allows you to monitor and capture events in an instance of SQL Server. With this information, you can use SQL Profiler to discover which queries are run most often, consume the most resources, and are causing bottlenecks. SQL Profiler can capture the events to either a table or a file so that you use the data to understand what is happening on your SQL Server. Either way, you can comb through the data with various tools and replay the events later. Replaying them allows you to analyze step by step what happened at a given instant, allowing you to correct problems and remove bottlenecks.

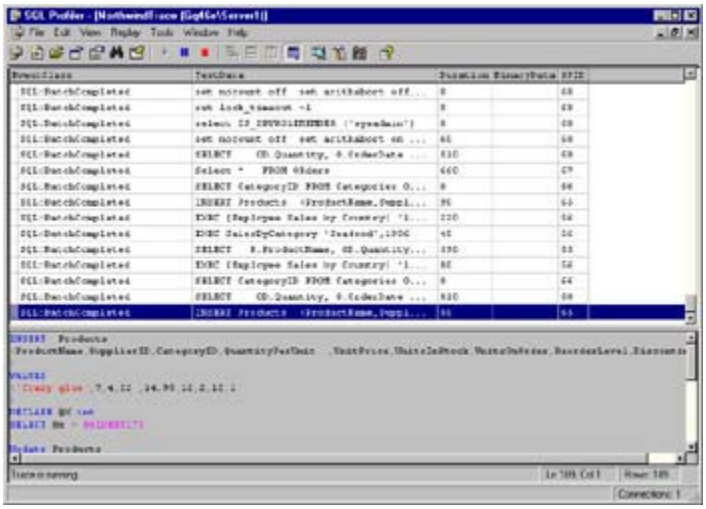


Figure 14.4: SQL Profiler with a trace running.

If you captured every event that happened on your SQL Server, you would have a vast amount of data that would be so large as to almost not be useful. Because of this unwieldy size, SQL Profiler allows you to filter the type of data of the trace by selecting the events and filters. An *event* is an action initiated within SQL Server that indicates something has happened. For instance, the initiation and the completion of a stored procedure execution are both events. Events are logically organized into *event classes* . Stored procedure initiation and completion are both in the stored procedures event class, along with events that describe stored procedure caching, compilation, and the like. By selecting specific events, you can create a trace designed to help you hone specific areas of your database performance.

Filters allow you to include and exclude events by establishing criteria for inclusion and exclusion in the trace. For instance, a filter can be created to limit the trace to a single user, database, or application.

Events and filters determine what records are recorded in the trace, but you can also select the information that is part of the record in a trace by selecting the data columns that you want to log. Not all columns are important in every trace; if you are tuning for speed, do you necessarily need information that is concerned with security?

Right about now you're saying, "Events? Event classes? Filters? Which ones do I choose? I just want to make my stored procedures run better." Because all this power can be overwhelming, several templates are included in SQL Profiler to help you quickly set up traces that will help you solve problems. If you are interested in tuning stored procedures with problems, you can choose the SQLProfilerTSQL_Duration template. If you want to identify queries that run a long time, you can choose the SQLProfilerTSQL_Duration template. In addition to choosing from many templates that already exist, you can create your own. You choose which template you want to use when you create a trace on the Trace Properties dialog box, as illustrated in Figure 14.5 .

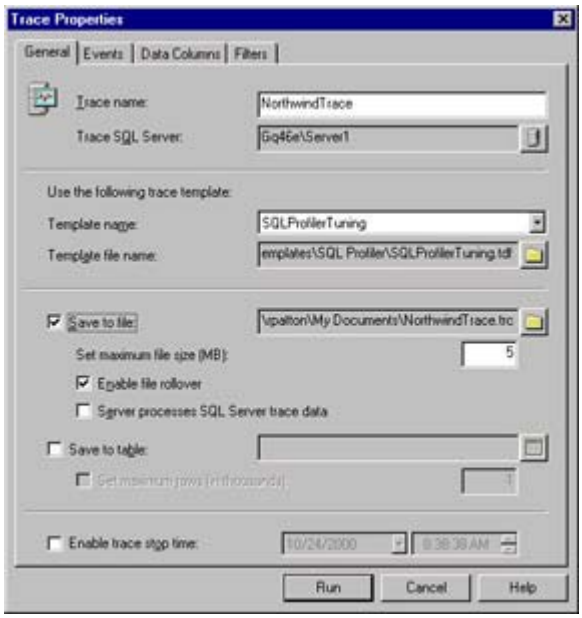


Figure 14.5: The Trace Properties dialog box General tab.

You should be able to see the enormous opportunity to use this tool to help tune performance of an application by creating traces specific to the application and traces that are looking for large amounts of resource use. Using this tool, you will be able to identify the worst-performing queries in your system and improve them systematically.

Exercise 1: Setting Up a Trace with SQL Profiler for the Northwind Database

In SQL Enterprise Manager, go to the Tools menu and select the SQL Profiler menu item. Then follow these steps:

1. In the SQL Profiler application, go to the File menu and choose New, and then select Trace from the pop-out menu.
2. In the Connect to SQL Server dialog box, enter the SQL Server name and the authentication information.
3. Once the information is authenticated, you will see the Trace Properties dialog box, as shown in Figure 14.6 . You should enter **NorthwindTrace** for the trace name and **SQLProfilerTuning** for the template name.

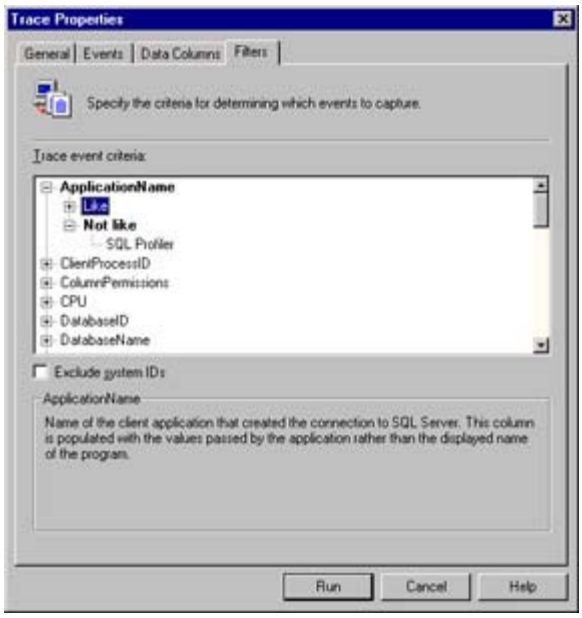


Figure 14.6: The Trace Properties dialog box Filters tab.

4. Next, click Save to file, and the File dialog box will prompt you for a name under which to save your trace. Enter **NorthwindTrace.trc** , and click Save.
5. Review the options selected on the Events and Data Columns tabs, and then go to the Filters tab, as shown in Figure 14.6 .
6. Navigate down the "Trace event criteria" tree to Database Name, and click the symbol.
7. Click the symbol next to Like in the tree, and enter **Northwind** . Then click Run.

Index Tuning Wizard

The trace files created by the SQL Profiler contain a great deal of information that you can comb through and search for ways to enhance the performance of queries in your database. The Index Tuning Wizard is a very powerful tool that can take the trace information created by SQL Profiler and use it to help you choose the optimal indexes for the loads that your server handles. It is able to look at the workload that you provide and suggest optimal clustered and nonclustered indexes based on its optimization algorithms.

As powerful as the Index Tuning Wizard is, it is only as good as its input. To use it effectively, you need to feed it the type of workload that you want to perform better. Do you want to optimize that database for your daily workload? Is it more important to tune it for your database for the heavy load that you have for a small window of time each day? Or is it more important to tune your system for a series of queries that generate reports for your CIO, who does not like to wait? Each of these scenarios could produce a different recommendation, and it is possible that one set of recommendations could negatively affect one of the others. You should carefully consider the type of load you want to analyze and inspect the recommendations carefully, using your knowledge of indexes to understand what adding and altering indexes will do to your system.

Exercise 2: Loading SQL Profiler trace file into the Index Tuning Wizard

1. In SQL Enterprise Manager, go to the Tools menu and select Wizards. In the Wizard dialog box, navigate to the Index Tuning Wizard item under Management, and click the OK button.
2. When the Wizard comes up, click the Next button.
3. Select Northwind in the database list box, and click the Next button.
4. Under Workload, choose "My Workload file."
5. In the File dialog box, navigate to the NorthwindTrace.trc file you created in Exercise 1 and then click the Open button.
6. Click the Select All Tables button, and then click the Next button.
7. Wait for the dialog box to state that analysis of the indexes based on your workload is in progress. Once it's finished, you will see the index recommendation that the wizard has come up with and an estimate of the performance increase that these recommendations will bring to your system. After reviewing them, click the Next button.
8. Choose "Save to script file" and, in the file dialog box, enter a filename to save your changes to. Click the Save button.
9. Click the Next button, and then click the Finish button.

SQL Query Analyzer

In SQL Profiler, you set up a trace to isolate the longest-running queries; now that you see the list, you would like to find out how or whether you can improve them. SQL Query Analyzer is the tool that allows you to complete this task. Among other things, it can allow you to trace the events of a query batch, as you can in SQL Profiler; inspect the Client Statistics; run the Index Tuning Wizard; and most important, look at the actual and projected execution plans for query batches.

Inspecting the execution plans might make it obvious why a query is consuming so many resources, but often it does not. You might have to try many variations of the same statements that are logically correct but use different execution plans to increase the performance of the query. Trying these successive iterations, you will be in good company. In his attempt to invent the light bulb, Thomas Edison said, "I have not failed. I've just found 10,000 ways that won't work." SQL Query Analyzer is the tool that lets us experiment with variations of queries to improve performance, because it gives us an intuitive graphical representation of our execution plans and presents us with quantitative cost results for the query. With this information, we are able to continually improve our queries. The empirical data that SQL Query Analyzer provide us is a graphical display showing useful information about the steps that the query took and about resources used at each step. These quantitative data are either the execution plan that the query used or an estimated execution plan that the query would use. The value of using the estimated execution plan is that you can tune queries without running them. This would allow you to tune a query that consumes a great deal of time rather quickly, compared with running it in successive iterations and waiting for it to finish.

Tips for Writing Better Queries

Before we use the Query Analyzer to tune queries, we should take a second to talk about how to write better queries by design.

One strategy you can use to help optimize your SELECT queries is to reduce the amount of information that they return. If you do not need rows, filter them out. Never use a SELECT * unless you need every column. Even then, it is bad practice because columns that you do not need could be added over time. Enumerate the columns that you *do* need, and you will reduce the amount of data that SQL Server must collect, process, and return. In addition, having fewer columns will allow SQL Server to take advantage of covered indexes, potentially providing much better performance. If you do not need to know the number of rows affected, you should include the following statement in your query batches:

```
SET NOCOUNT ON
```

If you don't turn this option on each time a SQL statement is executed, the number of rows affected will be sent to the client. If you don't need it, you shouldn't send it.

In addition to limiting the amount of data you return, you can improve your query performance by increasing the likelihood of reusing execution plans. If your query engine cannot find an execution plan that it can use, it must take the time to create one. Prior to version 7.0, SQL Server used to precompile stored procedures to reduce the amount of work it had to do in creating execution plans. Starting with version 7.0, SQL Server records the execution plan for each SQL statement for comparison against future SQL statements. This granular approach increases the likelihood that a matching execution plan can be used across stored procedures.

If you cannot use a stored procedure, it is best to use parameterized queries. Like stored procedures, they are easier for the query engine to find existing execution plans. All your database objects in your queries should use fully qualified names because that helps the matching process when you are looking for execution plans to reuse. Finally, you should not name your stored procedures starting with "sp_" unless it is in the master data-base, because the query engine will always look in the master database for it first. If you create a stored procedure that starts with "sp_" in the Northwind database, SQL Server will always look for it in master database first. This adds time to your query, or worse, it could run a query in the master database with the same name but different statements.

In general, do *not* use query hints. In certain cases, using query hints might help SQL Server on a specific query, but it could prevent the query engine from choosing an even better plan as indexes are added or statistics are updated. In addition, the SQL Server query engine is an ever-improving black box, and each revision and upgrade could make using that query hint much slower than the plan the query engine would have come up with itself. If you find an instance of a very important query that can be improved with the use of a hint, you could consider it, but you should enumerate those queries and test their performance against versions that do not use the query hints over time.

Finally, you should avoid using cursors if at all possible. In general, cursors consume a great deal of resources and are very slow. This slowness is due to the fact the cursors operate on a set of data on a row-by-row basis, taking a long time to process the set and consuming extra memory while it exists. If you can do the same processing with a temp table using a table type variable or client side, you should. If you must use a cursor, try to select the best type. If you need to only read the results, make sure you use a read-only cursor. If you need to perform updates, choose an optimistic one. In addition, choosing a forward-only cursor also reduce the burden on your server by consuming fewer resources to process the rows.

Query Execution Plan

The easiest way to understand how to tune a query with Query Analyzer is to step through it. We want a query that will return to us all the CustomerIDs for all the orders in the Northwind database that are either IN or WA. First, you need to open Query Analyzer and connect to the Northwind database. Next, go to the Query menu and select the Show Execution Plan menu item. Once you have done that, type our first attempt at the query, which follows, into Query Analyzer and run it:

```
SELECT      *
FROM        Orders          O,
            Customers       C
WHERE       ShipRegion =    'WA'
AND         O.CustomerID = C.CustomerID

UNION

SELECT      *
FROM        Orders          O,
            Customers       C
WHERE       ShipRegion =    'ID'
AND         O.CustomerID = C.CustomerID
ORDER BY C.CustomerID DESC
```

As you can see in Figure 14.7 , the execution plan results are displayed in the bottom half of the query screen on the Execution Plan tab.

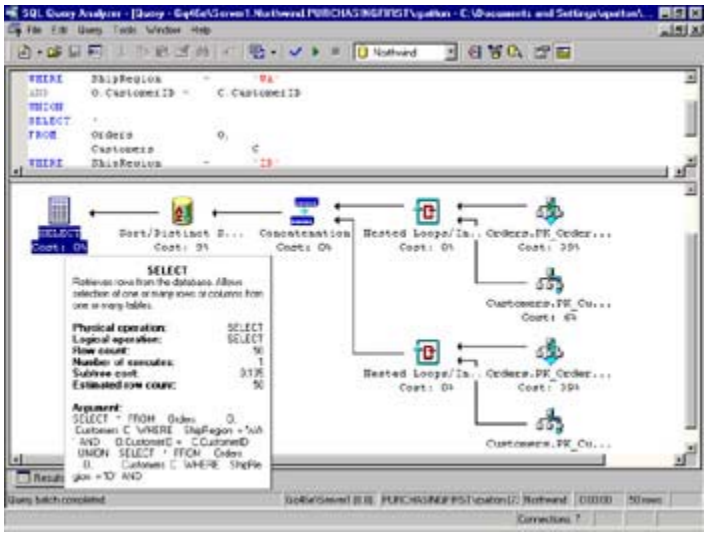


Figure 14.7: An execution plan in Query Analyzer.

If you select the Execution Plan tab, you will see the graphical flow-chart with symbols and percentages representing the query's execution plan. If you move your mouse onto the icon to the farthest left that has the word SELECT, as dem-onstrated in Figure 14.7 , a box will appear under the icon, showing your query and a subtree cost of 0.135. That is the value of all the parts that it takes to create your query added together, and this is the value that you will seek to reduce.

The first thing that you will notice is that there are two subtrees to the farthest right, each accounting for 39 percent of the cost of the query. If we could get rid of the UNION, we could remove one of the branches and hopefully improve our query performance results. Try the following query statement:

```
SELECT      *
FROM        Orders          O,
            Customers       C
WHERE       (ShipRegion =   'WA'
OR          ShipRegion =    'ID')
AND         O.CustomerID = C.CustomerID
ORDER BY C.CustomerID DESC
```

If you look at your execution plan now (see Figure 14.8), it will look much simpler, with fewer steps. If you mouse-over the SELECT icon, you will notice that the subtree cost has decreased to 0.0759, an improvement over our previous results. In addition, the display shows that 69% of the query's resources have been allocated to a clustered index scan on the Orders table. We should try to reduce that amount. Perhaps if we replace the * with the only columns that we need, a different, more efficient index can be used. Let's try this:

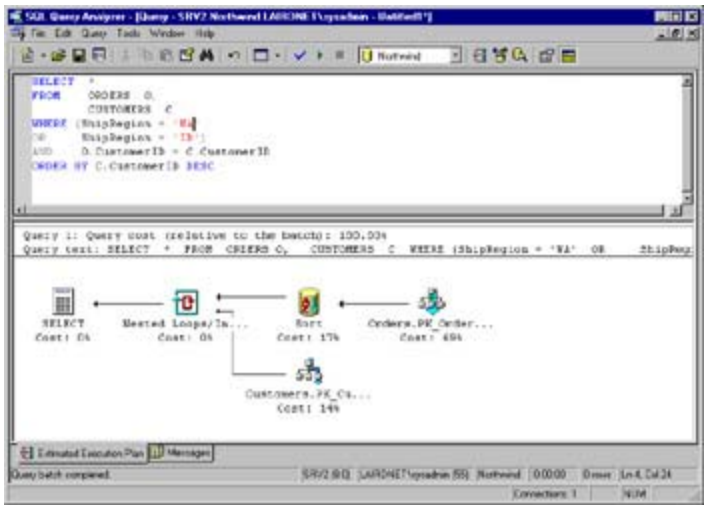


Figure 14.8: The next step of our execution plan in Query Analyzer.

```
SELECT      C.CustomerID
FROM        Orders          O,
            Customers       C
WHERE       (ShipRegion =   'WA'
OR          ShipRegion =    'ID')
AND         O.CustomerID =  C.CustomerID
ORDER BY C.CustomerID DESCWe
```

We can use the Index Tuning Wizard to further analyze our query and determine if SQL Server can recommend potential indexes. To do this, you must:

1. Highlight your query.
2. Choose the Index Tuning Wizard menu item on the Query menu.
3. Click the Next button.
4. Choose Thorough, and click the Next button.
5. Choose SQL Query Analyzer Selection, and click the Next button.
6. Click the Orders table, click the Next button, and wait for the processing screen to disappear. You will see a series of indexes but only one new one, named Orders9. It is an index on CustomerID and ShipRegion. Click the Next button.
7. Choose Apply changes, and choose Execute recommendations now. Then click the Next button.
8. Click Finish.

If you run the query again, you will see a new execution plan. Inspect it and you will see that the cost will be 0.0290, which is much lower than it was. The query took advantage of the covered index that you built.

A word of caution is in order here: You should not build an index for each query. Carefully choose which queries are most important to run well, and balance those choices against the impact of maintaining those indexes.

Optimizing Server Performance

In addition to creating a well-designed database schema, efficient queries, and indexes, the hardware and software configuration of the server are important components of optimizing your database's performance. This is true because SQL Server is merely an application that runs on top of an operating system that is running on a series of hardware components. Each of these—SQL Server, the operating system, and the hardware—can be optimized for the tasks that your database applications will be expected to accomplish.

A database server is inherently a data repository, and many of the important elements of tuning a database server involve tuning its random access memory (RAM) utilization and physical disk subsystem. In order to maximize the performance of your database, you need a basic understanding of the hardware available to you, the configuration options you can set in SQL Server and in the operating system, and how to use the performance-monitoring applications to spot existing and potential bottlenecks.

Hardware Configuration

The hardware you choose and how you use it are very important factors in a database server's performance. The faster the data can be moved to the client from disk or from the client to disk, the faster the database. Of course, different components in different configurations can give you different results, so it is important to understand the resources that are available and how you can use them to maximize your throughput. In addition to understanding the hardware itself, you need to understand what your SQL is designed to do, because that affects the resources you will need.

If your system is designed to be a decision support system (DSS), should you spend more money on adding processors, or would that money be best spent on adding disks? What if your system is going to be an online transaction processing (OLTP) system? If your database server will extensively use full-text searching, how will that affect your resource allocation? In order to answer these questions, you need to understand how SQL Server uses the physical resources that are available to it. With this knowledge, you should be able to design the hardware architecture you need to satisfy your requirements and avoid performance bottlenecks.

Processors

When people talk about the power of their servers, they often start with the speed of their CPUs. The amount of megahertz (millions of cycles per second) of your processors is not the whole story.

In addition to the amount of megahertz, the amount of cache memory on each CPU can greatly affect its throughput. The more cache a CPU has, the less wait time there is to read and write to the main memory, because the information can be stored in the much faster cache, allowing for much greater throughput and shorter waits.

You should also keep in mind that SQL Server is designed to take advantage of multiple processors. Symmetric multiprocessor support allows SQL Server to execute queries simultaneously or in parallel. In general, CPU performance is more critical for DSS systems than for OLTP because aggregations, sorting, and massive joins are more CPU-intensive than are data reads and writes that are typical of OLTP systems. In addition, you should keep in mind that populating full-text catalogs is very processor-intensive.

Memory

RAM is often an area that offers significant performance improvements for most applications, including SQL Server and Windows 2000. RAM provides the working memory of your system; by having a great deal RAM, you avoid having to access the much slower disk arrays. More RAM is always better, but systems that run parallel queries and populate full-text indexes (and most are primarily DSS systems) require even more RAM.

If the more RAM, the better, how much can you get? Windows 2000 Server is a 32-bit operating system; as a consequence, it can address only 4GB of memory. Windows 2000 Advanced Server and Data Center are also 32-bit operating systems, but through the Address Windowing Extensions (AWE) API, it can address 8GB and 64GB, respectively. You need at least 64MB of RAM for SQL Server, plus 32MB of RAM for the operating system. The more RAM you have, the more data that can be stored in the data cache and the more procedures that can be stored in the procedure cache. If the data and procedures that you are querying are in RAM, the server will not have to visit its disk system, providing a substantial improvement in performance.

Disks

No one will discount the importance of RAM, but it very important to note that the most common hardware bottleneck for database systems is related to disk input and output (I/O). That might seem somewhat intuitive, because after all, a database is about its data. However, even in simple data transactions, many more files are involved than the files that contain the source data, and all of them are stored and retrieved from the disk system. Among the most important files you should be aware of when trying to optimize your database server are these:

- Table files
- Nonclustered index files
- Transaction log files
- Full-text indexes
- TempDB database files
- Windows operating system files
- Windows paging files

One of the most fundamental things you need to choose when you first start optimizing your database server is the type of disk system to use. Most often, the only appropriate

choice is a redundant array of independent disks (RAID). Defined in 1987, RAID not only provides better performance, but it offers fault tolerance. When choosing a RAID solution, you need to balance your needs for data redundancy, speed, and cost.

RAID0

RAID0 uses striping without parity, providing the fastest possible read and write times. Striping data spreads it out across multiple disks so that the system can write and read from the multiple disks simultaneously, providing greater throughput. Since the data is striped across multiple disks, the failure of any single disk will result in the loss of data. A RAID0 system comprising five 8GB disks would provide 40GB of storage.

Tip A hardware-based RAID system is much faster than a software-based RAID system. A software-based system must use system resources to manage the RAID operations. These resources are not available for the operating system or applications, including SQL Server, so server performance is not optimal. Many hardware-based RAID controllers also provide large disk cache resources, increasing the performance of disk read and write operations.

RAID1

RAID1 provides redundancy by implementing *disk mirroring* . A RAID0 system is a system of two disks, one containing a copy of the first; if one fails, the other takes over and there has been no loss of data. A RAID1 system consisting of two 8GB disks would provide 8MB of storage. RAID1 does not use data striping and as a consequence it is not faster than a single-disk system.

RAID5

RAID5 stripes the data as RAID0 does, but RAID5 provides redundancy through striping extra parity information. These extra data are used to recover from failure of any one disk. Because of this additional information, RAID5 does not write information as fast as RAID0 or RAID0+1, but it does read information nearly as fast. A RAID5 system requires an extra disk, and with it, parity information is striped across all disks. A RAID5 system with five 8GB disks can store only 32GB of data: (5 - 1) * 8GB = 32GB. Each disk would have 1.6GB parity information and 6.4GB of data. A system with six 8GB disks could store only 40GB of data: (6 - 1) * 8GB = 40GB.

RAID0+1

RAID0+1 systems use striping without parity, providing the fastest possible read and write times and redundancy by implementing disk mirroring. RAID0+1 is a RAID0 disk-striping system that has had RAID1 disk mirroring applied for redundancy. Whereas RAID5 increases cost a bit, RAID0+1 increases cost a great deal: In order to provide 40GB of storage with 8GB hard drives, you would need 10 hard drives—five for the striping and five more to mirror the disks that have been striped!

Placing Files

As you can see in Table 14.2 , you can useRAID0 disk arrays for files that do not require fault tolerance—in other words, for information that can be rebuilt—but they require optimal performance such as the Windows paging file, TempDB database, and full-text indexes. It might be nice to have these files on a system that provides fault tolerance, but it is not essential. The Windows paging file and TempDB will be rebuilt each time the computer is rebooted. All the data that a full-text index represents are stored in the database tables, and the index can be rebuilt at any time. If you want to preserve the speed but absolutely require the fault tolerance, you can place those files on RAID0+1 disk arrays, but your cost will go up significantly due to the disk drive requirements to support mirroring.

Table 14.2: Common RAID Levels

Level	Description	Fault Tolerance	Read Performance	Write Performance	Cost
No RAID	Single disk	No	Normal	Normal	Inexpensive
RAID0	Striping without parity	No	Fast	Fast	Expensive
RAID1	Mirroring	Yes	Normal	Normal	Moderate
RAID5	Striping with parity	Yes	Fast	Slower	Expensive
RAID0+1	Striping with mirroring	Yes	Fast	Fast	Most expensive

If you do require fault tolerance but can sacrifice some performance, or if you cannot afford the cost, you can use RAID5 disks. Typically, the files that contain the data in tables and transaction logs are stored on RAID5 disk arrays. They are common, affordable, fast at writing information (though not as fast as RAID0 or RAID0+1), very fast at reading information, and, most important, fault tolerant. Of course, if you can spare the expense, you can also achieve fault tolerance with greater speed using the more expensive RAID0+1 systems.

In addition to the disk arrays themselves, the number and use of I/O controllers and data channels also contribute to the success of data access. Similarly to hardware versus software RAID systems, a good I/O controller should have its own processing ability so that it does not burden the general-purpose CPUs that will be needed to handle SQL Server operations. Generally speaking, SCSI controllers are good enough, but a Fibre Channel controller will give you the best performance at a higher price. Fibre Channel controllers can sustain a throughput of 100MBps (there are plans increase the throughput to 200MBps and 400MBps), and SCSI can sustain a throughput of only approximately 40MBps. Each controller has a number of channels that it can use to “talk” to its disks.

Obviously, the best scenario is to use one channel to one disk array. Likewise, the lower the ratio of I/O controllers to disk arrays, the better your overall performance.

Each controller is important to the success of your system, and ideally you would place each on its own RAID disk arrays, but often that is not possible. Commonly, we choose how to distribute these files among a fixed set of disks. To do that effectively, we must know something of our system. If you can, you should place the tables that carry heavy traffic on different controllers as their indexes. This will allow a degree of parallelism for both reads and writes. If you insert a row or delete a row from a table, each nonfull-text index must be implemented in the transaction. If the indexes are on the same disk system, they will be done serially. However, if they are on different disk systems, the indexes could be updated in parallel to updating the table. In this example, there are other factors to consider, too. Most updates and deletes are logged operations, and that information will need to be written to the transaction log. If the transaction log is on a different RAID controller, this operation too can be done in parallel.

Likewise, you can see that queries that utilize a great deal of TempDB or the paging file should be placed on different disk arrays and controllers than those that contain their source data. If possible, you should place the operating system on its own disks that are fault tolerant, but they do not have to be particularly fast.

Software Configuration

In addition to configuring your hardware in an optimal manner, you can choose to optimize the throughput of your database through certain software-based settings in SQL Server and the operating system. The values you set could determine whether you can get the desired performance out of the hardware that you have chosen.

SQL Server Settings

Since SQL Server is your application, it would stand to reason that within SQL Server are some things that you can configure to optimize your database server (see Figure 14.9). First and foremost among the settings you can choose for your database are the memory settings. SQL Server tunes its own memory usage, taking what it needs and releasing the rest to the operating system, but it will stay within a range you set with these parameters:

- Min server memory
- Max server memory

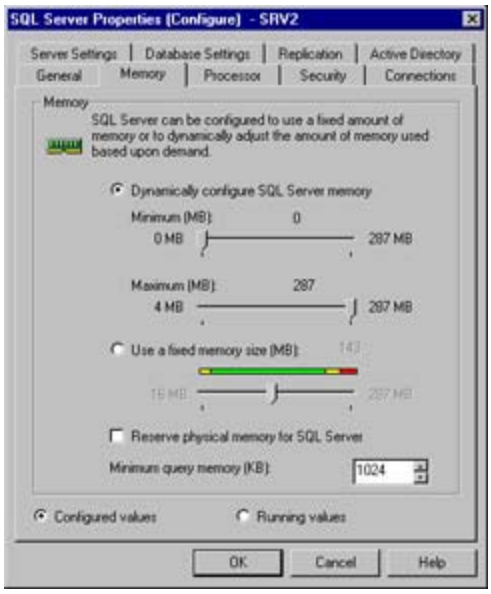


Figure 14.9: SQL Server Properties.

Max server memory is very important if you are using full-text searching to any great extent, because you need to limit the amount of memory SQL Server will use so as not to starve the MSSearch service. It is recommended that you limit SQL Server to half of the available memory (physical plus virtual) if you are running full-text searches. Otherwise, it is best to figure out how much the operating system needs and then set SQL Server to use all the available memory minus the memory that the operating systems needs. If you are running other applications on your SQL Server, you will have to account for them, too, in a similar manner, but you really should not be running other applications on your database servers.

In addition to the memory settings, you can also configure parameters that affect I/O activity with the *recovery interval parameter* **.** This parameter can affect how often SQL Server writes modified pages to disk, called a *checkpoint* . The default value is 0, which means that the data should be written to the database in 1 minute or less, depending on the current activity in the database. By increasing the value of the parameter, you can increase overall throughput by making SQL Server write more information in less frequent bursts. This is often desirable when your disk approaches 100 percent utilization. You can alter the value of the recovery interval in either Transact-SQL or the Query Analyzer using the sp_configure stored procedure and the following syntax:

```
EXEC sp_configure 'recovery interval', 1
RECONFIGURE WITH OVERRIDE
```

It is important to note that this value can be set for each database. You can increase the value in the Northwind database, but leave it at the default in the Foodmart database. You can also change it with Enterprise Manager like this:

1. Right-click on the server on which you want to make changes, and then click the Properties menu item.

2. In the SQL Server properties dialog box, click the Database Settings tab.
3. Under Recovery, in the Recovery interval (min) box, enter an integer to set the maximum amount of minutes that SQL Server could spend recovering each database at startup.

Altering the recovery interval in Enterprise Manager applies it in each of the databases on the server, whereas running it in the query window alters it in only that database.

If your system is on OLTP system that has multiple processors, you might want to turn off parallelism. If you don't, the optimizer will check to see if it can run the query in parallel for each and every query, which should be a very rare occurrence in an OLTP system. By applying your knowledge that the system is unlikely to run queries in parallel, you can the reduce the processing and memory overhead, helping improve the performance of your overall system.

We have talked a great deal about indexes and their importance to a well-performing database. Indexes need to be defragmented over time. You can check the fragmentation level of the indexes on a table by using DBCC SHOWCONTIG in the Query Analyzer. You can view the statistics for all the indexes on a table with a command like this:

```
DBCC SHOWCONTIG(tablename) WITH ALL_INDEXES
```

DBCC SHOWCONTIG will show a great deal of information, but the Avg. Bytes Free per page, Avg. Page density (full), Logical Scan Fragmenta-tion, and Extent Scan Fragmentation can indicate how fragmented the index is. If your index is fragmented, the Avg. Bytes Free per page will be a high number, and the Avg. Page density (full) will be low. Likewise, high numbers (values over 10 percent) for Logical Scan Fragmentation and Extent Scan Fragmentation indicate fragmentation unless the table is heap (a *heap* is a table that does not have a clustered index).You can defragment an index with either DBCC INDEXDEFRAG or DBCC REINDEX. DBCC INDEXDEFRAG does not hold long-term locks and, in that respect, can be better for production systems. DBCC REINDEX does hold locks, but it is much faster if your index has a great deal of fragmentation, whereas DBCC INDEXDEFRAG is faster with a small degree of fragmentation. You can run the DBCC INDEXDEFRAG command to defragment an index using the following syntax:

```
DBCC INDEXDEFRAG(database, tablename, indexname)
```

You can run DBCC REINDEX to accomplish the same task:

```
DBCC DBREINDEX (tablename, 'indexname')
```

Or you could use DBCC DBREINDEX to defragment all indexes in a table:

```
DBCC DBREINDEX (tablename)
```

Keeping the statistics for your indexes up to date is very important for the query engine to create optimal execution plans. You can ensure that this is done by setting the Auto Update Statistics database option. Generally, this is the best option. However, some systems suffer extreme loads during certain periods of time. For those systems, you might want to turn the Auto Update Statistics database option off and have them updated on a set schedule that takes advantage of known periods of time when there are more resources to spare. Likewise, you can turn the Auto Shrink database option off to prevent SQL Server from checking every 30 minutes to see if the database files can be reduced. This is unneeded overhead on very active servers. Once again, you can schedule a periodic shrinking of the database for times when you know the load is not as severe.

You can also tune performance in your SQL Server by setting values for the maximum number of connections. User connections by default are dynamically tuned; in most cases, you should leave them that way. However, if you want to limit the number of connections, you can set a maximum value. This value can prevent your server from being overwhelmed by too many connections. However, if you do set this value, you must keep in mind that the server will allocate 40KB for each connection, whether it is used or not. You can set this value to 100 with the sp_configure like this:

```
EXEC sp_configure 'user connections',100
```

You can accomplish the same task in Enterprise Manager by following these steps:

1. Right-click the server name. In the pop-up menu, click the Properties item.
2. In the SQK Server properties dialog box, click the Connections tab.
3. In the Maximum concurrent user connections text box, enter a cardinal value as a threshold.

Tangential to setting the user connections is the ability to set the maximum worker threads. If you have enough memory available, you can set it to a value that should allow you to have one thread per connection. That value would be equal to the maximum number of user connections, plus five. If you set a fixed value for the maximum number of connections, it is trivial to come up with the number. If you are letting SQL Server determine the number of connections, you need to monitor the performance of the server to discover a reasonable maximum value. If you lower the number for the default of 255, you will free memory; if you increase the number, you will consume more memory. Therefore, it important to make sure that you have the memory available to use. You can set this value to 105 to accommodate the connection with the sp_configure, like this:

```
EXEC sp_configure 'max worker threads',105
```

You can accomplish the same task in Enterprise Manager by following these steps:

1. Right-click the server name, and in the pop-up menu, click the Properties item.
2. In the SQK Server properties dialog box, click the Processor tab.
3. In the Maximum worker threads text box, enter a cardinal value greater than 32 as a threshold.

Finally, you can configure the query governor option to limit the number of seconds that queries can consume before they are terminated. Doing so prevents individual queries from hogging all the resources in the system, which would starve all queries. Be aware that the query governor uses an estimated plan for those queries, so they would not start if it appeared that they would exceed the value that you set. In the Query Analyzer, you set the limit for an individual connection like this:

```
SET QUERY_GOVERNOR_COST_LIMIT 10
```

That code would change it only for your connection. If you wanted to change it for all connections, you would need to use the sp_configure stored procedure, like this:

```
sp_configure 'query governor cost limit', 10
  GO
  RECONFIGURE
```

You can change the query governor cost limit only when it shows that the advanced options parameter is set to 1. In addition to being able to set with the Transact-SQL in the Query Analyzer, you can also change it in Enterprise Manager like this:

1. Right-click the server name, and in the pop-up menu, click the Properties item.
2. In the SQL Server properties dialog box, click the Server Settings tab.
3. Under Server behavior, select or clear the "Use query governor to prevent queries exceeding specified cost" check box.
4. In the text box next to that check box, enter a cardinal value as a threshold.

Windows 2000 Settings

Just as it is important to set parameters in SQL Server to help our database reach its full potential, we should also configure some settings in the operating system.

Experience tells us that one of the first things we need to do is set the amount of virtual memory available to the server. On most servers, the virtual memory should be 1.5 times the amount of the physical memory on the server. However, if you are running full-text searches, you need to increase the memory, because full-text indexing is memory-intensive. You need to set the memory to three times the amount of the physical memory. If you do not expect to perform many full-text searches or you expect to conduct them over a very trivial set of data, you might need to experiment to discover if you can set the value of the virtual memory to less three times the amount of the physical memory. In addition to setting the virtual memory amount higher, you need to make sure that MS Search will be allowed to use some of that memory.

In addition to making the correct virtual memory settings, you also need to set a few parameters for Windows 2000 to get the most out of your SQL Server. Although it is set by default when SQL Server 2000 is installed, you need to make sure that Maximize Throughput for File Sharing is set. You can set this value by following these instructions:

1. On your SQL Server, click Control Panel under Settings from the Start menu.
2. Double-click the Network and Dial-Up Connections Control Panel.
3. Right-click the Local Area Connection item, and then click the Properties menu item.
4. In the Local Area Connection Properties dialog box, click File and Print Sharing for Microsoft Networks in the Components list.
5. Click the Properties button underneath the Components list.
6. In the File and Print Sharing for Microsoft Network Properties dialog box, choose "Maximize data throughput for network applications."
7. Click the OK button in the File and Print Sharing for Microsoft Network Properties dialog box.
8. Click the OK button in the Local Area Connection Properties dialog box.

In order to get the best performance out of your SQL Server, you should run no other applications, including Windows 2000 components, on it. You should not allow your SQL Server to be used as a DHCP, DNS, or domain controller, if it all possible. Furthermore, do not allow your disk systems to use data file encryption. The time that it takes to encrypt and decrypt the information can burden your processors.

Finally, in order to achieve the best performance, you should set the server to optimize performance for background services at the expense of foreground applications. This does not give the SQL Server prominence over the foreground applications but instead puts them on equal footing. You can do this by:

Configuring optimization for background services:

1. On the Start menu, choose Control Panel under Settings.
2. 2. Double-click the icon for Server.
3. Under Optimize Performance, choose Background Services.

Performance Monitor

For many configuration options that help optimize your SQL Server, no hard and fast rules dictate the exact amount of RAM, type of disks, or number of processors required. You might believe that you have chosen the ideal combination of hardware, laid out your files correctly, and configured SQL Server and Windows 2000 to run optimally, but you cannot know for sure without empirical data.

An operating system tool called Performance (formerly known as Performance Monitor) can provide you with the data you need to ensure that you have the right resources in the right quantities. Performance provides the practical data that allows you to see how the resources of SQL Server are being used. As a consequence, Performance is a critical tool needed to tune your server over time by helping you isolate bottlenecks and examine resource use.

Performance is a Microsoft Management Console (MMC) snap-in that, as illustrated in Figure 14.10 , allows you to collect data from your servers about all aspects of their performance. The Performance application uses counters provided by several objects available to it, including ones that allow you to collect information on your disks, memory, CPU, and even SQL Server itself. You can use Performance to view the data in real time, or you can have it monitor in the background and save the data to a file so you can do a detailed analysis of it.

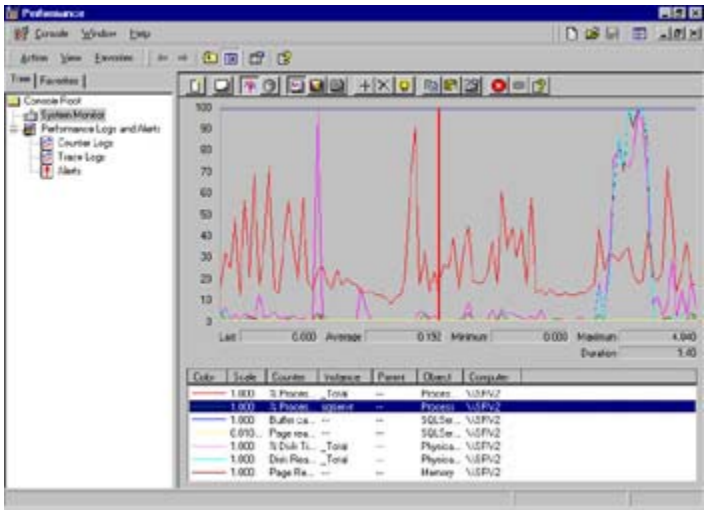


Figure 14.10: The Microsoft Management Console snap-in.

In order to use the tool effectively, you must choose counters that are indicative of your system's performance. An obvious component of a well-functioning system is not having your processors "maxed out." You can use counters from the System object to spot bottlenecks with regard to your CPU:

- System: % Total Processor Time
- System: Processor Queue Length
- System: Context Switches/sec

% Total Processor Time can be used to show the utilization percentage across each processor. Processor Queue Length is used to show the number of processes waiting for the CPU at any given time. A % Total Processor Time greater than 90 percent or a Processor Queue Length greater than 2 is indicative of a bottleneck at your CPU. In such an event, you need to get either more processors or more powerful processors to handle the workload.

Another counter available from the System object is Context Switches/ sec. This counter keeps track of the number of times a processor had to switch from one thread to another. If the Context Switches/sec counter is greater than 500 and your Processor Queue counter is high, you could be suffering from a CPU bottleneck. You can try to counteract this problem using the lightweight pooling option, which will switch from a thread-based scheduling model for CPU time to a fiber-based scheduling model. You can enable

lightweight pooling by using the sp_configure stored procedure in the Query Analyzer, like this:

```
sp_configure 'lightweight pooling', 1
```

Several counters are available to spot problems associate with memory utilization. In particular, the counters associated with the Memory object that are beneficial to monitor are:

- Memory: Pages/Sec
- Memory: Page Reads/Sec
- Memory: Available Bytes

Pages/Sec tells you how many times you had to retrieve from or write to disk due to page faults; in a well-performing system, this counter should be 0. Anything greater is indicative of a bottleneck. Likewise, a Page Reads/Sec counter greater than 5 indicates how many times the system had to go to the paging file to gather the information needed by the processor. Available Bytes indicates how much physical memory is left. If it is less than 4096, you need more memory.

It is important to resolve your memory and processor bottlenecks before you monitor for disk bottlenecks. Processor or memory bottlenecks can mask a disk bottleneck because the system is going too slow to expose it. Once you verify that the memory and processors are not your source of trouble, you can look toward the disk system. As mentioned before, disks are the most common bottleneck in database systems, and the most important counters used to detect these bottlenecks are under the PhysicalDisk object:

- PhysicalDisk: % Disk Time
- PhysicalDisk: Avg. Disk Queue Length

% Disk Time is a measurement of the utilization of your disks. If % Disk Time exceeds 90 percent, your server is experiencing disk bottlenecks. Avg. Disk Queue Length is a measurement of requests waiting for disk access. If this counter exceeds two times the number of spindles in your disk array, your server is experiencing disk bottlenecks You should have one spindle per disk, so a RAID5 system that has five disks should not have a disk queue length greater than 10 (5 spindles * 2).

Sp_configure

As we talked about how to make the settings optimize SQL Server, we kept calling on a single stored procedure: sp_configure. sp_configure is the primary system-stored procedure to interface with the global configuration settings for the server; for that reason, we should take a minute to explore it. The syntax for sp_configure is:

```
sp_configure [ @configname], [ @configvalue]
```

@configname is a varchar with a maximum length of 35 that represents the option name to configure. @configvalue is an int that is the new value for the option. Both are optional, and if you omit both parameters, a listing of all the options, valid ranges, and values will is shown. If you omit @configvalue, you will display the current value of the option. If you include it, you set the value of the option to @configvalue. This statement shows you the current number of maximum worker threads:

```
EXEC sp_configure 'max worker threads'
```

This would set the maximum worker threads to 71:

```
EXEC sp_configure 'max worker threads',71
```

Many of the options, including the max worker threads, query governor cost limit, and user connections, require that you have already used the sp_configure to set the show advanced options to 1. You can do so like this:

```
EXEC sp_configure 'show advanced options', 1
GO
RECONFIGURE
```

You might have noticed in the preceding query batch and in previous query batches that the RECONFIGURE statement often follows the sp_configure execution. Although all options will be changed to their new values when the server reboots, some options such as the one immediately preceding can be forced to take place immediately. You can do this by running RECONFIGURE.



Since you most likely will have multiple disks arrays in your server, you can create an instance of the PhysicalDisk object for each disk in your system. If you attempt to isolate different functions on different disk arrays, it will be much easier to tune your disks, because you can see where the bottlenecks are and which disks have resources available. With that information, you might want to redistribute some of your files in your database system to help achieve maximum throughput. Note that to use this performance object, you must ensure that the Disk Performance Statistics Driver is enabled. You can do this by running diskperf –y from the command line, and then rebooting your server. This should be done only on your development servers, not your productions servers. Overhead is involved in using the disk counters, which would slow down your production servers if you were to enable the Disk Performance Statistics Driver.

Although Performance is part of an operating system, it can be used with SQL Server-specific objects and their counters to monitor and tune SQL Server. Among the counters that are the most important for tuning SQL Server are these from the SQL Server 2000 Buffer Manager, Memory Manager and Cache Manager objects:

- SQLServer: Buffer Manager: Buffer Cache Hit Ratio
- SQL Server: Cache Manager: Cache Size (pages)
- SQLServer: Buffer Manager: Total Pages
- SQL Server: Memory Manager Object: Total Server Memory (KB)

The Buffer Cache Hit Ratio counter indicates how often SQL Server satisfies data requests from the cache buffer compared with the number of cache lookups. If this ratio does not exceed 90 percent, you can increase the amount of RAM allocated to SQL Server to achieve the highest possible buffer cache hit ratio. The Cache Size is the number of 8k pages that are being used for the SQL Server data cache. Ideally, this should be most of the memory that is not being used by the operating system.

In addition to the these two counters, the User Connections counter on the SQL Server General Statistics object can be useful to quantify the amount of activity that is going on to give you perspective on the other counters. Finally, the Working Set under the Process object can be used to show the amount of memory that the SQL Server process is using. If the Total Server Memory is near either the min server memory or max server memory, you should adjust the server configuration parameters to increase or decrease the minimum or maximum memory settings.

Summary

You can optimize a database on many levels; it involves choosing the right options, hardware, and layout. It also involves using tools to gather empirical evidence on what is going on, so you can play to your strengths. SQL Query Analyzer, SQL Profiler, Performance, and the Index Tuning Wizard are all tools that you can use to improve various aspects of your database and server configuration.

SQL Query Analyzer offers you an interactive "what if?" tool to understand and test individual sets of queries. It also allows you to tie into the Index Tuning Wizard. The Index Tuning Wizard can analyze a query against the databases structures and give you suggestions as to how an index might help your query. The Index Tuning Wizard can also be used with the trace information provided by SQL Profiler. SQL Profiler can help you monitor all the events in your database. It does this by allowing you to choose the types of activity that you would like to monitor; with it, you can figure out which queries need to be tuned.

One of the tools that is part of Windows 2000 is the Performance application. It is a tool that helps you analyze how the hardware in your system is being used and what problems are related to that hardware. Performance can help you pinpoint problems with processors, memory, and disks. Despite the fact that it is an operating systems tool, Performance has the ability to discover SQL Server-specific problems in your server.

Although SQL Server is generally a self-tuning system, settings within it can allow you to increase the performance of your database. To leverage these settings effectively, you must understand the type of resources that your database needs. SQL Server is an application, and as such it must run on top of an operating system. You can configure several operating system options to help you maximize the throughput of your databases. Like SQL Server, the operating system must run on top of a set of hardware components. The type and quantity of hardware components that you choose can have a tremendous impact on your overall system perform-ance. In particular, pay attention to your RAM and disk-based memory; the choices that you make in regard to these components can have a great impact on the performance of a typical database system.

In addition to all the ways that you can tune your database's queries, indexes, hardware, and software, SQL Server 7.0 gives you a new option to handle extreme loads: federated database servers. A database federation is a series of servers that work in conjunction with each other, using distributed partitioned views. This structure allows you to not only scale up your database by increasing the power of you server; it allows you to add extra servers to scale out. If well thought and well laid out, this system can enable you to handle the most enormous workloads imaginable.

FAQs

Q. I want to understand better how to lay out my tables in a distributed partitioned view. Could you illustrate how I would partition the tables in Figure 14.11 across federated servers? Assume that my system is for an e-commerce application and that the most significant transactional volume occurs in the ordering tables, Orders and Order Details. In addition, the Customers table has a high degree of transactional volume, but not as much as the ordering tables. All the rest of the tables in the database are relatively static by comparison. Finally, assume that it is important to have a high degree of integrity for information in the Customers and Products tables.

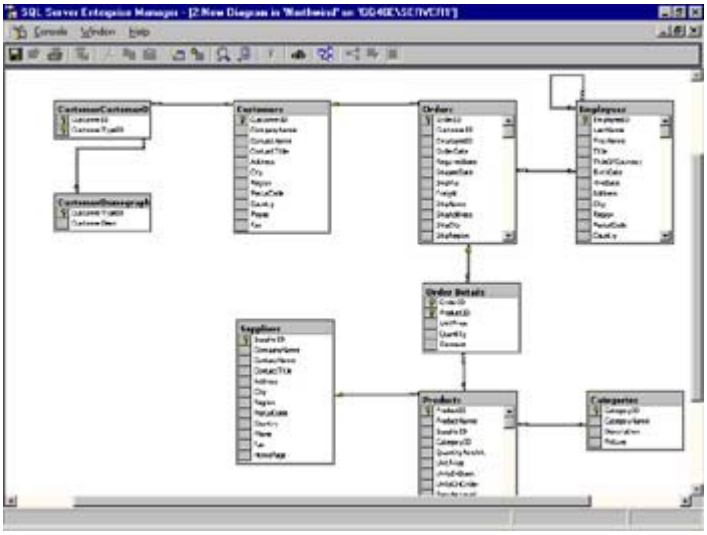


Figure 14.11: Data structures in the Northwind database that could become part of a distributed partitioned view.

A. The ordering tables are where all the activity is, and by creating a distributed partitioned view across the federation on Orders and Order Details, we can gain the most performance enhancement. The logical partition column for both would be OrderID. By making OrderID the partition column, we assure that Order Details records that are associated with Orders are on the same server. This will reduce interserver communication. The rest of the tables can be cloned to each server, and with the exception of Products and Customers, they can be kept synchronized through replication. The Products and Customers tables will require triggers in order to preserve the degree of integrity needed.

Q. I'd like to defrag all my indexes every night, when no one is accessing the database. How can I do this without scheduling each and every statement for every index?

A. You can create a stored procedure that will handle this task, and schedule it to run each night. The stored procedure that you need to create would query the system tables for a list of all user-defined tables. Then it would run DBCC DBREINDEX against them all. Your stored procedure would look something like this:

```
CREATE PROCEDURE Indexes_RebuildAll

AS
------------------------------------------------------------
```

```
--AUTHOR:     Robert A Patton
--DATE:       October 17, 2000
--COMMENTS:   This procedure will run DBREINDEX against
--            all user-defined tables within this database.
------------------------------------------------------------

DECLARE       @tablename   sysname
DECLARE       @temp     varchar(265)
--  List user defined tables
DECLARE      tnames_cursor   CURSOR FOR
                                    SELECT  name
                                    FROM   sysobjects
                                    WHERE  type = 'U'
OPEN tnames_cursor
     FETCH NEXT
     FROM   tnames_cursor
     INTO   @tablename
     WHILE  (@@fetch_status <> -1)
     BEGIN
       IF (@@fetch_status <> -2)
       BEGIN
            SELECT @temp = 'Updating '+ RTRIM(UPPER(@tablename))
            PRINT  @temp
            EXEC  ('DBCC DBREINDEX(['  + @tablename+'])' )
       END
  FETCH NEXT FROM tnames_cursor INTO @tablename
  END
  PRINT     '---------------------------------------'
  PRINT     'Indexes have been rebuilt for all tables.'
DEALLOCATE tnames_cursor
```

Q. What are the options that I can set with the sp_configure stored procedure and their valid values?

A. Tables 14.3 and 14.4 detail the options. They are grouped by ordinary and advanced options.

Table 14.3: Ordinary Configuration Options

Configuration	Default	Minimum	Maximum	Requires
Allow updates	0	0	1	No
Default	0	0	9999	No
Max text repl	65,536	0	2,147,483,647	No
Using nested	1	0	1	No
Remote access	1	0	1	Yes
Remote login	20	0	2,147,483,647	No
Remote proc	0	0	1	No

trans				
Remote query timeout	600	0	2,147,483,647	No
Show advanced options	0	0	1	No
Two-digit year cutoff	2049	1753	9999	No
User options	0	0	32,767	No

Table 14.4: Advanced Configuration Options

Configuration Option	**Default**	**Minimum**	**Maximum**	**Requires Restart**
Affinity mask	0	0	2,147,483,647	Yes
Awe enabled	0	0	1	Yes
C2 audit mode	0	0	1	Yes
Cost threshold for parallelism	5	0	327,675	No
Cursor threshold	-1	-1	2,147,483,647	No
Default full-text language	1033	0	2,147,483,647	No
Fill factor	0	0	100	No
Index create memory	0	704	2,147,483,647	No
Lightweight pooling	0	0	1	No
Locks	0	5000	2,147,483,647	Yes
Max degree of parallelism	0	0	32	No
Max server memory	2,147,483,647	4	2147483647	No
Max worker threads	255	32	32767	Yes
Media retention	0	0	365	Yes
Min memory per query	1024	512	2147483647	No
Min server memory	0	0	2147483647	No
Using nested	1	0	1	No

triggers				
Network packet size	4096	512	65,536	No
Open objects	0	0	2,147,483,647	Yes
Priority boost	0	0	1	Yes
Query governor cost limit	0	0	2,147,483,647	No
Query wait	-1	-1	2,147,483,647	No
Recovery interval	0	0	32,767	No
Scan for startup procs	0	0	1	Yes
Set working set size	0	0	1	Yes
User connections	0	0	32,767	Yes

Q. As illustrated in Figure 14.12 , I have a database server that has two RAID5 disk arrays, two RAID0 disk arrays, and one RAID0 disk array. It is an OLTP system that has heavy activity in two major tables that have high transaction volume, Sales and Session, and does not use full-text indexes. How should I organize the database files, indexes, TempDB, operating systems, and paging file to maximize disk throughput on my system?

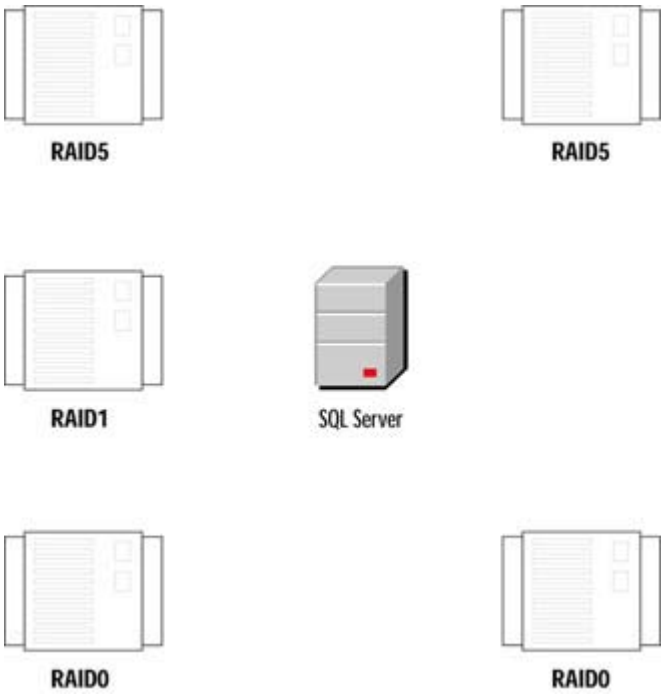


Figure 14.12: A database server with RAID5, RAID1, and RAID0 arrays.

A. You should place your database files on the RAID5 drives so that you have fault tolerance. Session should go on one drive and Sales should go on the other so that they are not competing for read/write time. You can place the TempDB and the virtual memory files on each of the RAID0 drives because they require the best performance possible and do not require fault tolerance. Finally, you can place the OS on the RAID1 array because it does not require the performance of RAID0 and RAID5, but it does need fault tolerance.

Index

Numbers

1NF. *See* First normal form
2NF. *See* Second normal form
3NF. *See* Third normal form
24 x 7 access, 314
24 x 7 applications, 296 , 297 , 303

A

Accent marks, 68
Access, granting. *See* Databases
Access | Read-only, 164
Access | Restrict Access, 164
Access license. *See* Client
Access (Microsoft), 415 , 551
Accounts Global Database Roles Permissions (AGDRP), 192
Accounts Global Local Permissions (AGLP), 192
ACID. *See* Atomicity Consistency Isolation Durability
Actions, 459
Active Directory, 203 , 207 , 249–260
analysis servers, registration, 261
analysis services, 260–261
computers, 263–265
databases, registration, 249
integration, 29 , 243 . *See also* Replication ; Windows 2000 Active Directory
FAQs, 294
publications
browsing/subscribing, 584–585
location/usage, 258–260
registration, 257–258 , 266 , 582–585
replication, 254–256
services, 6
SQL Server, registration, 246–248
SQL server objects, examination tools/techniques, 251–253
understanding, 245–246
users, 263–265
Active Directory | List this database in Active Directory, 167
Active Directory Service Interface (ADSI), 252 , 262
Active Server Pages (ASP), 209 , 483 , 556
Active Server Pages+ (ASP+), 6
Active users, 495
Active/active mode, 50–52
Active/passive mode, 49–50

ActiveX
data objects, 6 , 539–542
scripts, 414 , 415 , 417 , 420
snapshot control, 549
ActiveX Data Objects (ADO), 500 , 633–639
Recordset object, 539
usage, 540
ActiveX Data Objects Multidimensional (ADO MD), 491
object library, 466
Add-ins, integration, 461
Address Windowing Extensions (AWE), 675
Addressing. *See* File Transfer Protocol
Administration
access, 194–197
integration, 243
FAQs, 294
tasks, automation, 290–293
Administration Wizard, delegation, 267
Administrative access. *See* Operating system administrative access
Administrative tasks, programming, 639–645
Administrator, password assignation. *See* System
ADO. *See* ActiveX Data Objects
ADO MD. *See* ActiveX Data Objects Multidimensional
ADSI. *See* Active Directory Service Interface
Advanced server, 104–105
AFTER, 616–617
triggers, 614 , 616 , 617
AGDRP. *See* Accounts Global Database Roles Permissions
Agents. *See* Distribution ; Log Reader Agent ; Merge Agent ; Queue Reader Agent ; Replication ; Snapshot Agent
processing, control/limitation, 588
Aggregate data, 260
Aggregation, 478
AGLP. *See* Accounts Global Local Permissions
Alerts, 266 , 271 , 290–293
defining, 292–293
Algorithms. *See* Data mining
ALL parameter, 653
ALTER DATABASE
command, 133 , 170
statement, 171 , 626
ALTER DATABASE...COLLATE statement, 160
ALTER PROCEDURE statement, 609
American National Standards Institute (ANSI)
dialect, 599
padding, 657

SQL, usage timing, 628–629
Analysis Manager, 462–463
Analysis Server, 21–30 , 461–462
accessibility, 21
installation, 262
registration. *See* Active Directory
Analysis Services, 20–23 , 644 . *See also* Structured Query Language ; Structured Query Language Server
accessing. *See* World Wide Web
architecture, 461–466
database, creation, 474–475
first-time installation, 467–468
IIS configuration, 494
improvements, 459–461
installation, 466–469
integration, 363
Processing Task, 27
programming, 643–645
requirements, 467
security, 492–493
implementation, 493
unattended installations, 473
upgrading, 468–469
Analysis Services Processing, 411
Anonymous subscriptions, 581
ANSI. *See* American National Standards Institute
Answers, 378
files. *See* Automated installations
API. *See* Application Programming Interface
Application
considerations, 553–559
requirements, determination, 45
roles, 228–231
Application Center 2000, 6
Application Programming Interface (API), 288 , 290
Application service provider (ASP), 58
Architecture. *See* Analysis Services ; Log shipping ; Structured Query Language ; System requirements. *See* Processor
Array functions, 483
Article, 547–548
AS400, 551
ASP. *See* Active Server Pages ; Application service provider
Atomic field, 181
Atomicity Consistency Isolation Durability (ACID)
properties, 564
rules, 565

Attributes, 491 , 502 . *See also* IDENTITY
values, 253
Authentication, 202 . *See also* Users ; Windows NT
mode. *See* Structured Query Language ; Windows authentication ; Windows-only authentication mode
Authoring. *See* Enhanced authoring ; Graphical authoring
object model, 363
AUTO mode, 522–524
Automated installations, answer file, 67
Availability. *See* Structured Query Language Server 2000
AVG aggregate function, 619
AWE. *See* Address Windowing Extensions

B

Background population, 401
Background services configuration, optimization, 683
Backup. *See* Integrated backup ; Merge replication ; Monthly backup ; Northwind database ; Snapshot files ; Snapshot replication ; Transactional replication
actions. *See* Distributor ; Publisher
device, 324
files, 307
extension, 340
storage, 340
history, 348
media retention. *See* Default backup media retention
number, 348
performing, 174–175
timing, 591–593
plan, specification. *See* Databases ; Transaction logs
power, providing, 101
retention period. *See* Complete backups ; Differential backups
schemes, sample, 315–316
sets
information, reading, 348
restoration, 348
subset selection, 348
storage, 304 , 307–317
media, 311–313
strategy, 297 . *See also* Remote backup strategies ; Replication
options, 302–304
plan/implementation, 296–319
selection, 302–307
testing, 354–355
tests, performing, 338
tools/techniques, 319–322

Backup | Last database backup, 160
Backup | Last transaction log backup, 160
BACKUP command, 322
Backup DB permission, 167
Backup disk directory, specification, 175–176 . *See also* Transaction logs
Backup Log permission, 167
Backup-and-restore model, 303
Backup-and-restore strategies, 308–311
Backup/restore process, 353
Backward compatibility. *See* Structured Query Language Server
BAK, 340
BCP. *See* Bulk Copy Program
BEGIN statement, 610
Benchmarks. *See* Transaction Processing Council
Bigint (data type), 600–604
Binary columns, 619
BizTalk Server 2000, 6
Biztalk.org, 510
BOL. *See* Books Online
Books Online (BOL), 30 , 467
Boolean ersult, 519
Boolean responses, 471
Bottlenecks, 665 , 685
Browsing. *See* Active Directory
ability, 549
Buffer Manager, 687
Bulk Copy Program (BCP), 409 , 433–440 , 471 , 634
mode, usage, 587
usage, 436–440
BULK INSERT
command, 444–446
example, 616
statement, 418
usage, 445–446
BulkCopy. *See* Structured Query Language Distributed Management Object
Object, usage, 440–444
Bulk-logged recovery, 165 , 444
Business orders, 298
Buyers, 367

C

C++, 579 . *See also* Visual C ++
header file, 288
C2 certification, 193–194
C2 configuration, 195

CA. *See* Certificate Authority
Cache Manager, 687
Calculated cells, 464
Capacity
limits. *See* Structured Query Language Server 2000
planning, 148–150
CASCADE option, 613
Cascading updates/deletes, 613–614
Case sensitivity, 68
CatalogName, 388
Certificate Authority (CA), 238
Challenge phrase, 203
Change-tracking population, 401
Changing dimension type, 22
Character data, 502
Character width, 68
CHECK constraints, 112 , 656
Check constraints, 556–559
CHECKDB, 289
Transact-SQL statement, 338
Checkpoint, 306 , 679
Cisco router, 571
Client
access license, 62–63
applications, access, 460
installation, 262
network utility, 87–88
Clients, 367
Cluster
failure, recovery. *See* Fail-over
services, 5 , 6
support, configuration, 75–76
Cluster Service
component, installation. *See* Windows 2000
configuration, 105
domain login account, 102
Clustered indexes, 138–139 , 661–665
Clustering, 23 , 458 , 486 . *See also* Structured Query Language Server ; Structured Query Language Server 2000
architecture. *See* Structured Query Language
capabilities/requirements, 100–101
planning. *See* Structured Query Language Server
upgrading. *See* Structured Query Language
Collation
options, 68–69
support, 19–20

Collections, 289
Column-level conflict tracking, contrast. *See* Row-level conflict tracking
Column-level merge tracking, usage, 588
Columns, 131 . *See also* Binary columns ; Data mining ; IDENTITY ; Partition column
enabling. *See* Full-text search
indexes, 555
removal. *See* Computed columns ; Derived columns
storage requirements. *See* Variable-length column storage requirements
XML data mapping. *See* Databases
COM. *See* Component Object Model
COM+. *See* Component Object Model Plus
Command-line utilities, 633–634
Comment, 502
Commerce Server 2000, 6
Communications security. *See* Network
Compatibility | Level, 166–167
Complete backups, 316
retention period, 307
Component Object Model (COM), 5 , 598
COM-based language, 289
component, creation, 411
interface, 466 , 622
object, 361 , 419
model, 643
tools, 361
Component Object Model Plus (COM+), 5 , 6 , 579 , 637
applications, 577
clusters, 96
Component services, 273
Compound primary key, usage, 555
COMPUTE, 619
COMPUTE BY, 619
COMPUTE BY clause, 525
Computed columns, removal, 184
Conflicts. *See* Replication
minimization, 585 . *See also* Subscribers
resolution, 577–579
tracking, contrast. *See* Row-level conflict tracking
viewing, 577
Conformed dimensions, 480
Connection
address. *See* Network
bandwidth/availability, 553
speed. *See* Physical Internet connection speed
Connections, 415
tab, 82–83

Constraints, 131
CONTAINS, 393 , 395–398 , 619
predicate, 381 , 395 , 396
CONTAINSTABLE, 393 , 395–398
predicate, 385
return table, 386
Context node, 507
Context switches, 684
Copy Database Wizard, 278–280
COUNT(*) functions, 619
Covered indexes, 664–665
CPUs, idleness, 327–328
Create Database Backup Wizard, 319–320
CREATE DATABASE statement, 128 , 161 , 170
Create Database Wizard, 168
completion, 158
usage, 153–158
Create Default permission, 167
Create Function permission, 167
CREATE INDEX statement, 123
Create Publication Wizard, 257 , 567–568
Create Rule permission, 167
Create SP permission, 167
CREATE TABLE
permission, 222
statement, 122 , 474 , 603
Create Table permission, 167
CREATE VIEW
operation, 631
permission, 222
statement, 122 , 525
Create View permission, 167
CRM. *See* Customer relationship management
Cross-server joins, 66
CSS, 504
CUBE, 619
Cube Wizard, usage, 475–479
CubeAnalyzer, 644
Cubes, 459 , 463–465 . *See also* Virtual cube
access. *See* HyperText Transfer Protocol
data security, usage, 479
design/construction, 475–480
editing, 479
elements. *See* Hidden cube elements
processing. *See* Enhanced cube processing
querying, 482–485

role, 492
structure, alteration, 466
technology. *See* Linked cube technology
Custom resolvers, constrast. *See* Default resolvers
Customer relationship management (CRM), 297 , 298
Customers, 367
Cyrix processor, 60

D

Daily (occurrence), 329
Data
access tools/technologies, 633–639
availability, cost, 298–299
changes, 303 , 564
channels, 677
connection, creation, 422
copying, 414 , 423
directory. *See* Default data directory
FAQs, 450–452
files, 137 . *See also* Primary data file ; Secondary data file
securing. *See* Structured Query Language Server 2000
format, 448
import/export, 407
method, performance considerations, 449
method choice, 447–449
tools, overview, 408–410
loading/transformation, DTS usage, 471–474
location, 552
loss, cost, 298–299
management, 414 , 423
manipulation tasks, 448–449
modification, 552–553
objects. *See* ActiveX
optimization information, updating, 172–174
pages, reorganization, 336
partitioning, 111 , 112
patterns, discovery, 491
publishing, 586
recovery requirements, determination, 296–299
reorganization, 172–173
replication. *See* Internet
required pages, calculation, 149
rows, size determination, 149
security, 197–201 , 493
usage. *See* Cubes

servers
configuration, 657–659
overview, 653–655
partitioning, 652–659
sources, 463
creation, 474–475
support. *See* Heterogeneous data sources
transformation, 414 , 423 . *See also* Data Transformation Service
types, 15–16 , 528 , 600–604 . *See also* String data type ; User-defined data type
updating/usage. *See* eXtensible Markup Language
warehouse
components, 470
design/construction, 469–474
populating, 470–471
warehousing, contrast. *See* Online analytical processing
Data Control Language (DCL), 624 , 631–633
Data Definition Language (DDL), 624–626
Data Manipulation Language (DML), 624 , 626–631
Data mining, 22–23419 , 454–459 . *See also* Structured Query Language Server
algorithms, 465 , 486
providers, 465
capabilities, 461
columns, 465
model
browser, 489–491
creation/editing, 486–488
usage, 488–491
multidimensional expressions, 491
OLE DB, usage, 491
Prediction Query Task, 27
solutions, 454
support, 466
training, 488–489
Data Transformation Service (DTS), 27–28 , 58 , 408–419
architecture, 412–416
changes/extensions, 48
Designer, 422–426
interface, DTS package execution, 430
Import/Export Wizard, 420–421
improvements, 410–412
performance considerations, 417–419
programming, 645–646
usage. *See* Data
Data Transformation Service (DTS) packages, 623
creation/editing, 419–429
execution, 429–433 . *See also* Structured Query Language ; Visual Basic

dtsrun.exe utility, usage, 429 , 430
dtsrunui.exe utility, usage, 432
file, saving, 428–429
saving, 427–429 . *See also* Meta Data Services ; Structured Query Language Server ; Visual Basic
security, 416–417
Database | Date created, 160
Database | Number of users, 160
Database | Owner, 159–160
Database | Size, 160
Database | Space available, 160
Database | Status, 159
Database Administrator (DBA), 263 , 265 , 267 , 290 , 495
Database Console Command (DBCC), 285–286
Database Console Commands (DBCC), 171
Database Copy Wizard, 58
Database Designer. *See* Structured Query Language Server
Database Maintenance Plan Wizard, 171–172 , 320–321
Databases, 130 , 303 , 345 . *See also* Differential databases ; Online transaction processing ; Relational databases ; Users
access, granting, 199–200
administrators, 180
training, 45–46
alteration, T-SQL usage, 168–171
attaching, 281–282
backup. *See* System
options, 304–307
performing, 323–334
plan, specification, 175
restoration, 344–348
showing, 345–347
changes
frequency, 297–298
reconstruction, 298
columns, XMLl data mapping, 529–534
configuration, 151 , 158–168 , 262
creation, 151–186 , 474–475 . *See also* Analysis Services
T-SQL, usage, 168–171
design. *See* Physical database
considerations, 553–559
modification, 585 , 588
performance tuning, 586
designers, 180
detaching, 281–282
enabling. *See* Full-text search
entities, identification, 180

FAQs, 240–242 , 595–596
files
deletion/creation, 302
growth, defining, 156
management, 140
naming, 155–156
unused space, removal, 173 , 336
growth rate, 307
ID, 400
integrity check, 174–175 , 338
location, specification, 155
maintenance
plan information, 133
plans, 286–288
tools, 284–290
management. *See* Structured Query Language
modeling, 145
tools, 180
name, 159
naming, 155
normalization, 181–184
options, 352–354
page, modification, 306
permissions, 206
properties, 249
registration, 266 . *See also* Active Directory
replication
design, 552–559
technique/configuration, 545
restoration, 344–345 . *See also* Replicated database
roles, 200–201 , 206 , 225–231 . *See also* Fixed database roles
usage, 463 , 480 , 492
scalability, 96–97
security, 189
implementation, 231–239
selection, 172
servers
inventorying, 45
partitioning. *See* Federated database servers
settings, 352–354
settings tab, 84–85
size/copying, 565
sizing, 148–149
solution, design, 144–145
subdirectory, creation, 340
tables

copying, 421
XML data mapping, 529–534
transferring, 410
usage, 463 , 480 , 492
users, 206 , 218–224
account addition, Transact-SQL usage, 220–221
adding, 219–221 . *See also* Enterprise Manager
virtual directory, creation. *See* Internet Information Server
Datacenter server, 104–105
Data-reduced schemas. *See* eXtensible Markup Language
DAT-DDS. *See* Digital audio tape-digital data storage
DB2, 551
DBA. *See* Database Administrator
DBCC. *See* Database Console Command ; Database Console Commands
DBCC CHECKDB command, 174 , 175
DBCC INDEXDEFRAG, 680
DC. *See* Domain Controller
DCL. *See* Data Control Language
DDL. *See* Data Definition Language
DDS. *See* Digital data storage
Decision Support Object (DSO), 466 , 643
Decision support system (DSS), 674
Decision trees, 23 , 458 , 486
Default backup media retention, 84
Default data directory, 84
Default instance, 68
Default log directory, 85
Default logins, 211
Default resolvers, custom resolvers (contrast), 579
Defaults, 131
DELETE
action, 131
query, 660
statement, 122 , 525 , 560 , 576
usage, 613–614 , 626–628 , 656
trigger, 614
Delimiters. *See* Field ; Rows
Denormalization, 457
contrast. *See* Normalization
DENY statement, 229 , 231 , 632
Denying (process), 211
Dependent dimension type, 22
Derived columns, removal, 184
Destination server, 115
Development libraries, 30
Devices, 348

Dewey decimal system, 661
Differential backups, 304–305 , 316
retention period, 307
scheduling intervals, 307
size/growth rate, 307
Differential database, 324
Differential Northwind database, 316
Digital audio tape-digital data storage (DAT-DDS), 311
Digital data storage (DDS), 311 , 312 . *See also* Digital audio tape-digital data storage
Digital linear tape (DLT), 311
Dimension table, 474
Dimensional table, 470
Dimensions, 261 , 460 . *See also* Conformed dimensions ; Parent/child dimensions ; Ragged dimensions ; Shared dimensions ; Virtual dimension
creation/editing, 480–482
defining, 480–482
functions, 483
understanding, 480
Disk unit, 300–301
Disks, 675–678
drive, utilization, 587
imaging support, 67
I/O throughput, 60
media, 311
space, 33 , 35 , 37 , 38
subsystem, 146–148
DISTINCT clauses, 619
Distributed Management Objects, 640–642 . *See also* Structured Query Language Distributed Management Object
Distributed partitioned views, 26 , 109–114
creation, 111–114
Distributed queries, 284
Distributed Transaction Coordinator (DTC), 76 , 77 , 83 , 273 . *See also* Microsoft Distributed Transaction Coordinator
Distributed view
creation, 111
usage/updating, 113–114
Distribution, frequency reduction, 587
Distribution Agent, 548 , 560 , 563 , 564
Distributor, 70 , 547 , 564 , 589 . *See also* Remote distributor
backup actions, 592–593
configuration, 573–574
information, 134
server, assignation, 565–566
strategy restoration. *See* Failed distributor
DLL. *See* Dynamic link library

DLT. *See* Digital linear tape
DML. *See* Data Manipulation Language
DNA, .NET influence, 9–10
DNS. *See* Domain Name System
Domain Controller (DC), 245 , 254–255 . *See also* Primary Domain Controller
Domain login account. *See* Cluster Service
Domain Name System (DNS), 246
Domain user account
requirements, 66
usage, 66
Drill-through, 465 , 492
Drives, usage. *See* Multiple drives
DROP DATABASE command, 170
DSO. *See* Decision Support Object
DSS. *See* Decision support system
DTC. *See* Distributed Transaction Coordinator
DTS. *See* Data Transformation Service
dtsrun.exe utility usage. *See* Data Transformation Service packages
dtsrunui.exe utility usage. *See* Data Transformation Service packages
Dynamic link library (DLL), 283 , 288
Dynamic partitioning, 553
Dynamic Properties Task, 27

E

Edit Recurring Job Schedule, 328
Electronic mail (E-mail)
addresses, 89
message, 290
notification, 266
ELEMENTS option, 524
E-mail. *See* Electronic mail
Encryption. *See* Multiprotocol encryption ; Secure Sockets Layer
END statement, 610
English Query, 30 , 359
application, 363–364
construction/deployment, 376–380
creation, 366–376
implementation. *See* World Wide Web
planning, 366–367
testing, 377–379
Development, 368
Domain, 368
FAQs, 404–405
improvements, 362–364
installation, 364–365

requirements, 365
Model, 373–376
Module, 368
OLAP Project, 371–373
overview, 360–364
Project, 368
project
components, 368–369
creation, 367–376
Regression, 368
run-time engine, 366 , 369
solution, 380
creation. *See* Web-based English Query
SQL Project, 369–370
creation, SQL Project Wizard usage, 370–371
Enhanced authoring, 362
Enhanced cube processing, 464
Enterprise Manager, 30 , 66 , 277–278 , 387 , 422 , 573–574 . *See also* Structured Query Language Server
database user accounts, adding, 220
DTS package execution. *See* Structured Query Language Enterprise Manager
fixed server role, login addition, 216–218
interface, 642
login, 216
SQL login, adding, 214–215
usage, 116 , 209 , 210 , 265
Windows logins, adding, 211–213
Enterprise resource planning (ERP), 225
Enterprise-level user, 207
Entities, creation/editing, 373–374
Entity attributes, defining, 180
Entity relationship (ER), 457
defining, 180
Entity-relationship diagrams, 180–184
Equality operators, 519
ER. *See* Entity relationship
ERP. *See* Enterprise resource planning
Error messages, transferring, 411
ESE. *See* Extensible Storage Engine
Excel (Microsoft), 448
Exchange Server, 551
version 5 . 5 , 5
version 2000, 6
EXEC permission, 222
Executable files, securing. *See* Structured Query Language Server 2000
EXECUTE

permission, 631
statement, 629
Execute Package, 410
Task, 27
EXIT command, 633
Expiration dates, 316
EXPLICIT mode, 522
Export. *See* Data
task, frequency, 448
Extended properties, 20
eXtensible Markup Language (XML), 483 , 499 , 500 . *See also* Select ... For XML
benefits, 501–502
data
mapping. *See* Databases
usage/updating, 534–542
data-reduced schemas, 505–506 , 526–529
definition, 500–501
documents, 502–504 , 506 , 542
FAQs, 544
limitations/syntax/usage. *See* FOR XML
overview, 500–510
Path Language (XPath), 500 , 502 , 506–509
context, 506
data types/conversions, 519–520
limitations. *See* Structure Query Language
queries, 12–13 , 518–521
overview, 518–519
usage, 520–521
specification, 519
query templates, execution, 516–518
resources. *See* World Wide Web
support, 10–14
support/limitations. *See* Structured Query Language Server
tags, 503
template, creation, 515–516
updategrams support, downloading. *See* Structured Query Language Server 2000
usage, 502–509
views, 12–13 , 525–534
example, 13
mapper. *See* Structured Query Language
XML-Data Reduced (XDR), 502
schemas, 12–13 , 500
example, 13
Extensible Storage Engine (ESE), 245
eXtensible Stylesheet Language (XSL), 500 , 502 , 504–505
style sheets, 515

F

Fact table, 470
Failed distributor, strategy restoration, 310–311
Failed publisher, strategy restoration, 310
Failed subscriber, strategy restoration, 310
Fail-over, 549
cluster failure, recovery, 110–111
cluster/server, 107
configuration, 262
Fail-over clustering, 26 , 72 . *See also* Structured Query Language Server ; Structured Query Language Server 2000
architecture. *See* Structured Query Language
implementation, 102–109
upgrading. *See* Structured Query Language
FASMI. *See* Fast Analysis Shared Multidimensional Information
Fast Analysis Shared Multidimensional Information (FASMI), 455–456
Fast Ethernet, 96 , 101 , 111
Fault tolerance, 146 , 152
FDDI, 96
Federated database servers
configuration, 657–659
overview, 653–655
partitioning, 652–659
Federated servers, 109–111
usage, 653
Fibre Channel controller, 677
Field. *See* Monotonically increasing fields ; Sequentially increasing fields
lengths, 433
prefix lengths, 433
storage types, 433
File autogrow, 150
File Transfer Protocol (FTP), 408–410 , 572
addressing, 575
information, specification, 573–574
paths, 573
Task, 27
usage. *See* Replicating ; Snapshot
Filed
delimiters, 433
FILEGROUP01 disk system, 152
FILEGROUP02 disk system, 152
Filegroups, 136–137 , 151 , 306–307 , 345
Files, 306–307 , 324 , 345 . *See also* Backup
filtering, 381
placement, 676–678

removal, 340
transferring. *See* Internet ; Network
Fill factor, 662
FILLFACTOR setting/property, 172
FILLFACTOR value, 336
Filtering. *See* Files ; Punctuation filtering
Filters, 381
option, 484
First backup to restore, 346
First normal form (1NF), 181
rules, complying, 183
Fixed database roles, 200
Fixed roles, 216–218 , 225–227
Fixed server role, login addition. *See* Enterprise Manager
Transact-SQL, usage, 218
Fixed-length fields, 149
FOR BROWSE clause, 525
FOR EACH clause, 542
FOR XML
clause, 515 , 522 , 534
limitations, 525
statement, 500
syntax/usage, 522–525
Free space, amount, 336 , 337
Free space per page percentage, changes, 336
FREETEXT, 363 , 393–395 , 619
predicate, 381 , 394
searches, 394
FREETEXTTABLE, 393–395
predicate, 385
return table, 386
FROM clause, 610
From device, 345
FROM statement, 484
FTP. *See* File Transfer Protocol
Full backup-and-restore model, 303–304
Full population, 401
Full recovery, 165
Full-process option, 489
Full-text catalogs
administration, 398–403
backup, 399–400
creation, 387–391
Full-text indexes, 382 , 675
administration, 398–403
construction, 391–392

creation. *See* Northwind database
performance considerations, 383–387
populating, 400–403
querying, 393–398
Full-text indexing, 275
Full-text Indexing Wizard, 387
Full-text query support, 363
Full-text search, 271 , 359 , 360 , 619 , 679
architecture, 382–383
column, enabling, 389–390
database, enabling, 388
enabling, 387–392
FAQs, 404–405
overview, 381–387
table, enabling, 388–389

G

GC. *See* Global Catalog
GFS. *See* Grandfather-father-son
Gigabit Ethernet, 111
Global Catalog (GC), 254
Global configuration settings, 686
Global groups, 192
Global subscription, 578
usage, 588
Globally unique identifier (GUID), 646
Goodwill, 298
Grandfather-father-son (GFS), 314
GRANT statement, 229 , 231
Granting (process), 211
Graphical authoring, 362
Graphical question builder, 362
GROUP BY clause, 123 , 619
Group policies, 246
Group-based administration model, 192
Groups, 492 . *See also* Filegroups ; Repeating groups
Growth planning, 148–150
Guest user account, 221
Guestpass, 234
GUI, 132 , 210
GUID. *See* Globally unique identifier

H

Hardware. *See* Replacement hardware

compatibility, 100–101
configuration, 674–678
failure, plan, 299–302
RAID, 64
requirements. *See* Structured Query Language Server 2000
upgrades, 586–587
upgrading. *See* Network ; Server
Hardware Compatibility List, 100
Hardware-based RAID system, 676
HAVING, 619
Heap, 139
Heterogeneous data sources, support, 550
Heterogeneous publishers, 551–552
Heterogeneous subscribers, 551–552
Hidden cube elements, 464
Hierarchy functions, 483
High-frequency tasks, 448
HKEY_LOCAL_MACHINE, 66
Host Integration Server 2000, 6
Hot swap, 64
HTML. *See* HyperText Markup Language
HTTP. *See* HyperText Transfer Protocol
HTTPS, 454 , 460 , 483
HyperText Markup Language (HTML), 415 , 510
documents, 381
parser, 503
table, 504 , 505
HyperText Transfer Protocol (HTTP), 454 , 460 , 501
access, 500
cube access, 483
query support, IIS configuration, 510–512
URL query support, 510–518
usage, 483 , 497 , 502 , 510 . *See also* Structured Query Language ; Structured Query Language Server ; Supported query methods

I

IBM DB2 6.x, 34
IDE. *See* Integrated development environment
IDENTITY, 440 , 654 , 657
attribute, 600 , 601
columns, 568 , 654
range, 555
Identity values
assignation. *See* Structured Query Language Server
management, 554–559

IIS. *See* Internet Information Server ; Internet Information Service
Image data, 135
image (data type), 657
Import task, frequency, 448
Import/export. *See* Data
job requirements, 447–448
Incremental population, 401
Index enhancements, 18–19
Index pages, reorganization, 172–173 , 336
Index rebuilds, scheduling, 402–403
Index Tuning Wizard, 668–669
SQL Profiler trace file, loading, 668–669
Indexed views, 16 , 121–124 , 617–622
creation, 124
requirements, 122–123
Indexes, 131 , 138–139 . *See also* Clustered indexes ; Columns ; Covered indexes ; Nonclustered indexes
deletion/creation, 302
performance considerations. *See* Full-text indexes
rules, 664–665
types, 661–665
understanding, 660
Informix, 628
version 7 .x, 34
Inline user-defined functions, 612–613
Input/output (I/O), 675
controller, 677
throughput. *See* Disks
INSERT
action, 131
operation, 662
permission, 222
statement, 113 , 122 , 525 , 560
usage, 576 , 614 , 626 , 656
INSTEAD OF, 614–616
trigger, 600 , 614–615 , 618 , 656
Integer values. *See* Single-column integer values
Integrated backup, 21
Integrated development environment (IDE), 63
Integrity
check. *See* Databases
enhancements. *See* Referential integrity enhancements
Interface adapters, 300
Internet
connection speed. *See* Physical Internet connection speed
data replication, 571–576

files, transferring, 587
standards support, 31–32
Internet connectivity, 466
Internet Information Server (IIS), 5 , 376 , 497
configuration. *See* Analysis Services ; HyperText Transfer Protocol
database virtual directory, creation, 511–512
server, 580
Internet Information Service (IIS), 5 , 6
Internet Protocol (IP)
addresses, 104 , 572 . *See also* Static IP addresses
subnets, 255 , 256
Internet Protocol Security (IPSec), 207 , 236 , 239
usage. *See* Windows 2000
Internet Security and Acceleration Server 2000, 6
Interprocess Communication (IPC) Mechanism, 236 , 311
Intersite replication, 255
Intranet, security measures, 571
Intrasite replication, 255
I/O. *See* Input/output
IP. *See* Internet Protocol
IPC. *See* Interprocess Communication
IPSec. *See* Internet Protocol Security
ISAPI, 515
DLL, 580
extensions, 512
.ISS file, recording, 75
IT department, 354

J

Jobs, 271
execution, 423
requirements. *See* Import/export
transferring, 411
Joins. *See* Cross-server joins
JScript, 417
scripts, 209

K

Kana character types, 68
KCC. *See* Knowledge Consistency Checker
Kerberos, 244
Key fields, 183 . *See also* Nonkey fields
Keys, 131
Knowledge Consistency Checker (KCC), 255 , 256

L

L2TP. *See* Layer Two Tunneling Protocol
LAN. *See* Local Area Network
Large-scale analysis, 456
Layer Two Tunneling Protocol (L2TP), 571
LDAP. *See* Lightweight Directory Access Protocol
LDP, 251 , 252
Lightweight Directory Access Protocol (LDAP), 251
LIKE searches, 363
Linked cube technology, 459
Linked servers, 282–284
creation, 111
Local Area Network (LAN), 255 , 408 , 483
link, 553
Location path, 507
Locking management, 140
Locks. *See* Processes/locks
Log file backups, 164
Log file write activity, 147
Log Reader Agent, 549 , 560
Log sequence number (LSN), 306
Log shipping, 27 , 102 , 115–121
architecture, 303
configurations, 172
information, 120
monitoring, 120–121
setup, 116–120
stopping, 119–120
support, 115
usage. *See* Transactional replication
Logical functions, 483
Login access, securing. *See* Structured Query Language Server
Logins, 210–216 . *See also* Default logins ; Enterprise Manager ; Remote logins ; Standard logins
accounts, 132
adding. *See* Windows logins
transferring, 411
Logs. *See* Transaction logs
monitoring information, 134
shipping configuration, 134
Log-shipping architecture, 303
Loops, implementation, 416
LSN. *See* Log sequence number

M

Magneto-optical disks, 307
Mail. *See* Structured Query Language
Maintenance | Collation name, 160–161
Maintenance | Maintenance plan, 160
Maintenance plan, 338–339
completion, 178–180
history, 178
Maintenance Plan Wizard, 172–180 , 286–288 . *See also* Database Maintenance Plan Wizard
MAKE DIRECTORY commands, 154
MAPI. *See* Messaging Application Programming Interface
Mapping schema, example, 13
Master database, 132 , 199 , 221 , 316 , 318
backup, 317
Master Stored Procedures, transferring, 411
MAX functions, 619
Max server memory, 679
MDC. *See* Meta Data Coalition
MDX. *See* Multidimensional expression
Measures
creation/editing, 480–482
defining, 480–482
understanding, 480
Media. *See* Disk media
families, 313–314
rotation plan, 307
sets, 313–314
Media rotation, 314
Member functions, 483
Member tables, 657
Memory, 33–38 , 674–675 , 685
amount, 392
requirements. *See* Structured Query Language Server
tab, 79
Merge Agent, 548–549 , 562 , 574
Merge replication, 550 , 561–563 , 565 , 577
backup, 591
enhancement process, 588
restoration, 591
Message queue, 410
Message Queue Server (MSMQ), 5 , 6
Message Queue Task, 27
Messaging Application Programming Interface (MAPI), 290
mail profile, 89
profile name, 90
Meta Data Coalition (MDC), 647

Meta Data Services, 20 , 273 , 598 , 622–623
DTS packages, saving, 428
Meta data services
programming, 647–648
Microsoft Cluster Service (MSCS), 100 , 101
setup. *See* NT 4.0 ; Windows 2000
Microsoft Distributed Transaction Coordinator (MS DTC) Service, 104 , 108 , 139 , 142–143
Microsoft Management Console (MMC), 265–274 , 462 , 684
snap-ins. *See* Structured Query Language Server
Microsoft resolvers, 577
Microsoft Search Service, 143 , 382–383
Microsoft SQL Desktop Engine (MSDE), 2
Microsoft.NET, 4–8
MIN functions, 619
Min server memory, 679
Mining. *See* Data mining
models, 463 , 465 , 485–486 . *See also* Online Analytical Processing ; Relational data mining models
role, 492
Mining Model Wizard, 487
MMC. *See* Microsoft Management Console
Mobile Information Server 2001, 6
Model database, 134
Modify Agent, 592
MOLAP, 22
Monitor server, 115
Monotonically increasing fields, 664
Monthly backup, 316
Monthly (occurrence), 329
MOVE statement, 276
MS DTC. *See* Microsoft Distributed Transaction Coordinator
MSCS. *See* Microsoft Cluster Service
MSDB, 199 , 647
Msdb database, 133–134 , 316 , 318
backup, 317
MSDE. *See* Microsoft SQL Desktop Engine ; Structured Query Language Server 2000 Desktop Engine
MSDN, 519
MSDN.Microsoft.com/XML, 510
MSMQ. *See* Message Queue Server
MS-SQL-OLAPServer, 24
MSSQLServerADHelper service, 143
MSSQLServerOLAPService, 144
MS-SQL-SQLDatabase, 24
MS-SQL-SQLPublication, 24

MS-SQL-SQLServer, 24
MSSSQLServer, 194
MSXML parser, 536
MTS, 83
Multidimensional analysis, 261
Multidimensional expression (MDX), 21 , 360 , 377 , 454 , 483–485 . *See also* Data mining
Builder, 454 , 461
Multidimensional OLAP (MOLAP), 478 , 479 , 496
Multimaster, 245
Multi-phase data pump, 411
Multiple drives, usage, 313–314
Multiple server instances, 68
support, 29–30
Multiple-instance support, 47
Multiprotocol encryption, 236–237
configuration, 236–237
Multivalue data, 181
MVS, 551

N

Name. *See* Databases
Named element, 502
Named pipe, 311
Namespaces, 642–643
National Security Administration (NSA), 193
Natural language restatement/answer. *See* Questions
.NET Enterprise Server, 3–8 , 254
.NET influence. *See* DNA
NetBEUI, 236
NetBIOS name, 102
Network, 301–302
adapters, setup, 103
communications security, 201–202 , 236–239
connection address, 30
files, transferring, 587
hardware, upgrading, 585
libraries tab, 86–87
load balancing, 5 , 6
utility. *See* Client ; Server
Networking, 33
Network-related tasks, 60
NO ACTION constraint, 613
Node test, 508
Nodes, 491

NOEXPAND, 620
Nonclustered indexes, 139 , 663–665
files, 675
Nonkey fields, 183 , 184
Nonlogged operations, performance, 302
NORECOVERY statement, 276
Normalization, 180 . *See also* Databases
denormalization, contrast, 470
Northwind database, 135 , 199 , 234 , 315 , 323 . *See also* Differential Northwind database
backup, 329 , 330
connection, 636
products table, full-text index creation, 390–391
trace setup, SQL Profiler usage, 667–668
usage, 511 , 530 , 671
NOT FOR REPLICATION flag, 554
setting, 556–559
NSA. *See* National Security Administration
NT 4.0, MSCS setup, 104
ntext (data), 135
ntext (data type), 633 , 657
NTFS, 103
NULL values, 524 , 608 , 656 , 657
Nullable column/expression, 619
Numerical functions, 484
NWLink, 236

O

Object
browser. *See* Query Analyzer
copy/paste functionality, 21
library. *See* ActiveX Data Objects Multidimensional
models. *See* Structured Query Language Distributed Management Object
security, 197–201
Object Linking and Embedding (OLE) DB, 6 , 409 , 634–639
data sources, 143 , 284
provider, 283 , 379 , 422 , 448 , 463 , 658
Subscriber, 587
support, 486
usage. *See* Data mining
version 2 . 5 , 425
ODBC. *See* Open Database Connectivity
ODER BY clause, 611
Office hours, 298
Offline storage, 307

Offsite storage, 315
OIM. *See* Open Information Model
OLAP. *See* Online Analytical Processing
OLE. *See* Object Linking and Embedding
OLTP. *See* Online transaction processing
On-demand snapshot execution. *See* Scripts
On-demand update, 402
One time (option), 328
Online Analytical Processing (OLAP), 128 , 260 , 454–459 , 643 . *See also* Multidimensional OLAP ; Real-time OLAP ; Relational OLAP
cube, 373
data
mining models, 486
warehousing, contrast, 455–458
enhancements, 21–22 , 459–461
levels, 374
Manager, 364
Mining Model Editor, 465 , 488
models, 464 , 488
OLTP, contrast, 455–458
Project. *See* English Query
Wizard, 362 , 363 , 372
properties, 374
services, 260
solution, 181 , 408 , 454 , 497
design/construction, 469–475
usage, 482–485
Online storage, 307
Online transaction processing (OLTP), 98 , 298 , 455 , 674
application requirement, 181
contrast. *See* Online analytical processing
data. *See* Time-limited OLTP data
databases, 457 , 470
system, 63 , 457 , 470 , 474 , 679 , 680
Open Database Connectivity (ODBC), 6 , 415 , 598 , 633–639
connection, 573
drivers, 475
Subscriber, 587
Open Enterprise Manager, 196
Open Information Model (OIM), 598 , 622 , 647
OPENQUERY functions, 619
OPENROWSET functions, 619
OPENXML, 12
statement. *See* Transact-SQL
Operating system administrative access, 233
Operational procedure, 316

Operators, 271 , 290–293
setup, 292
Oracle, 412 , 448 , 551 , 628
capabilities, 415
version 7 .x/8.x, 34 , 411
Oracle database support, 363
Organizational Unit, 246 , 263
OSQL, 634
OSQL/ISQL tools, 209
OU. *See* Organizational Unit
OUTER joins, 619
Overwrite media, 326

P

Packages, 413
creation, 419–420
reusing, 420
security. *See* Data Transformation Service
workflow, 415–416
Pages
reorganization, 336 . *See also* Data ; Index pages
split, 662
Parallel data pump data, 410
Parent/child dimensions, 460 , 481
type, 22
Parity information, 147
Parsed term, 395
Partition column, 657
PartitionAnalyzer, 644 , 645
Partitioned views, 657 . *See also* Distributed partitioned views
Partitioning. *See* Data ; Dynamic partitioning
goal, 111
Passive mode. *See* Active/passive mode
Passwords, 203
assignation. *See* System
transmission. *See* Structured Query Language
Path. *See* eXtensible Markup Language ; File Transfer Protocol ; Location path ; Relative paths
Path Language. *See* eXtensible Markup Language
PDC. *See* Primary Domain Controller
Performance, 396
considerations. *See* Data ; Data Transformation Service ; Full-text indexes ; Replication
enhancement process. *See* Merge replication ; Snapshot replication ; Transactional replication
FAQs, 689–693

monitor, 683–687
optimization, 495 . *See also* Queries
tuning, 495 , 585 . *See also* Databases ; Replication
tools/techniques, 651
PerlScript, 417
Permissions, 192 . *See also* Databases
assignation. *See* Users
assignment, 262
Per-snapshot execution. *See* Scripts
Physical database, design, 146–150
Physical disk, 685
Physical Internet connection speed, 571
Physical storage architecture, 135–139
PING, 103
PivotTable service, 466
Point in time restore, 346
Point-to-Point Tunneling Protocol (PPTP), 236 , 571
Population. *See* Background population ; Change-tracking population ; Full population ; Incremental population
Post-snapshot execution. *See* Scripts
Power Point presentations, 381
PPTP. *See* Point-to-Point Tunneling Protocol
Precedence constraints, 415
Prefix lengths. *See* Field
Primary data file, 137
PRIMARY disk systems, 152
Primary Domain Controller (PDC), 245
PRIMARY filegroup, 136 , 137
Primary key, 657
Primary.mdf, 306
Privileges, assignation. *See* Serverwide operations
Processes/locks, 132
Processing instruction, 502
Processor
architecture requirements, 33 , 35 , 36–38
license, 62
licensing model, 34
queue length, 684 , 685
tab, 79–81
Processors, 674
Production environment
migration, implementation, 44
monitoring/optimization, 44
Production migration, testing, 44
Products table, full-text index creation. *See* Northwind database
Profiler. *See* Structured Query Language

Progress 8.x/9.x, 34
Project Wizard. *See* Structured Query Language
Properties, 347
Proxy Server, usage. *See* Replicating
Publications, 547–548 , 569
automatic synchronization, 310
configuration, 574–575
creation, 262 , 566–568
registration/location/usage. *See* Active Directory
subscribing ability. *See* Replication
updating, 587
Publish-and-subscribe
environments, 308–311
system, 303
Publisher, 71 , 546–547 , 569 , 588–589 . *See also* Heterogeneous publishers
backup actions, 592
configuration, 573–574
restoration strategy. *See* Failed publishers
Publishing. *See* Data
enabling. *See* Server
Pubs database, 134–135 , 199
Pull Subscription Wizard, 584
Pull subscriptions, 548
contrast. *See* Push subscriptions
Punctuation filtering, 396
Purchasers, 367
Push subscriptions, 548
pull subscriptions, contrast, 569–570

Q

Queries, 377 , 378 . *See also* DELETE ; Distributed queries ; SELECT ; Structured Query Language ; Subqueries ; UPDATE ; XPath queries
data graph, 495
execution plan, 671–673
frequency, 495
governor option, 681
hour graph, 495
parameters, specification, 517–518
performance, 667
optimization, 659–665
response graph, 495
results, 377
run time, 495
speeding, 585
templates, execution. *See* eXtensible Markup Language

writing tips, 669–670
Query Analyzer, 14–15 , 604–609 . *See also* Structured Query Language
Object Browser, 605–607
usage, 209 , 230 , 387
Query optimizer, usage, 173 , 336
Querying, 475 . *See also* Cubes ; Full-text indexes
HTTP usage. *See* Structured Query Language Server
Questions, 378
natural language answers, 377
natural language restatement, 377
Queue length. *See* Processor
Queue Reader Agent, 549
Queued updating, 549
subscribers, 29 , 561

R

Ragged dimensions, 460
type, 22
RAID. *See* Redundant Array of Inexpensive/Independent Disks
RAISERROR statement, 625
RAND, 618
Ranges, retrieval, 665
RAW mode, 522
RDBMS, 628
Read-only access property, 164
Read-only partitioning capability, 109
READTEXT statement, 630
Real-time OLAP, 464
RECONFIGURE statement, 686
Recovery. *See* Bulk-logged recovery ; Full recovery ; Simple recovery
interval, 84
parameter, 679
model
selection, 52
settings, options, 165
strategy, 297
creation, 317–319
plan/implementation, 296–319
testing, 354–355
Recovery | Model, 164–165
RECOVERY statement, 276
Redundancy, 257
configuration, 262
Redundant Array of Inexpensive/Independent Disks (RAID), 64 , 100 , 102 , 146 . *See also* Hardware ; Software

configurations, 148
controllers, 64 , 100 , 154 , 678
disk arrays, 677
levels, 147
RAID 6+, 64
RAID0, 64 , 148 , 383–384 , 675–676
RAID0+1, 153 , 383–384 , 676
RAID1, 64 , 102 , 152–153 , 301 , 383 , 586 , 676
RAID4, 64
RAID5, 102 , 383 , 676
fault tolerance, 384
system. *See* Hardware-based RAID system
REFERENCES permission, 612
Referential integrity enhancements, 17 , 613–614
Registry
keys, 30
rebuilding, 75
Regression features, 362
Relational data mining models, 485–486
Relational databases, 129–131
Relational Mining Model Editor, 465
Relational mining models, 488
Relational OLAP (ROLAP), 459 , 479 , 496
Relational operators, 519
Relationships, creation/editing, 374–376
Relative paths, 507
Remote backup strategies, 66
Remote distributor, 547
Remote logins, 132
Remote Procedure Call (RPC), 236
Repeating groups, 181
REPLACE statement, 276
Replacement hardware, 100
Replicated database, restoration, 591–594
Replicating
FTP, usage, 572–573
Proxy Server, usage, 571–572
VPN, usage, 571
Replication, 66 . *See also* Active Directory ; Intrasite replication ; Merge replication ; Schema change replication ; Snapshot replication ; Structured Query Language Server 2000 ; Transactional replication
Active Directory, integration, 582–585
agents, 548–549 , 592
architecture. *See* Structured Query Language Server
backup strategies, 588–594
compatibility, 550–552

conflicts, 575–576
design. *See* Databases
environments, 308–311
features. *See* Structured Query Language Server
information, 134
method, selection, 564–565
Monitor, 272 , 546
performance
considerations, 585–588
tuning, 586–587
programming, 646–647
publications, subscribing ability, 549
requirements, 552
scripting, 309
servers, upgrading, 51–52 , 70–71
services, 28–29 . *See also* Structured Query Language
support. *See* Structured Query Language Server 2000
system, 303
tab, 85
technique/configuration. *See* Databases
transaction. *See* Uncommitted replication transaction
types, 548
validation, 309
Replication Conflict Viewer, 576
Reports, generation, 177
Repository object, 647
Reserved keyword changes, 48
Resolvers, contrast. *See* Default resolvers
Resource management solution, 207
Restatements, 378
RESTORE command, 322 , 345 , 349–351
RESTORE FILELISTONLY, 351
RESTORE HEADERSONLY, 351
RESTORE LABELONLY, 351–352
Restore list, 347 , 348
RESTORE LOG, 276
Restore tools/techniques, 319–322
RESTORE VERIFYONLY, 352
Retention dates, 316
Retention period. *See* Complete backups ; Differential backups ; Snapshot files
RETURN statement, 610
RETURNS statement, 611
REVOKE statement, 229 , 231
Revoking (process), 211
ROLAP. *See* Relational OLAP

Roles, 492 . *See also* Application ; Cubes ; Databases ; Fixed roles ; Mining ; Standard roles ; User-defined roles
permissions, assignation, 235
ROLLUP, 619
Rowguid, 582
RowGuid, management. *See* Tables
ROWGUIDCOL
column, creation, 488
property, assignation, 555
Row-level conflict tracking, column-level conflict tracking (contrast), 577
Rows
calculation, 149
delimiter, 433
identifiers, identification, 180
size, determination. *See* Data
total number, 395
RPC. *See* Remote Procedure Call

S

SAN. *See* Storage area network ; System area network
Scalability. *See* Databases ; Structured Query Language Server ; Structured Query Language Server 2000
enhancements, 25
Scalar user-defined functions, 610
Scaling, up/down, 94–98
Scaling up/out, 652
Schedule, specification, 173–174
Scheduling intervals. *See* Differential backups ; Transaction logs
Schema, type, 520
Schema change replication, 550
Scripts. *See* ActiveX ; JScript
on-demand execution, 550
per-snapshot execution, 550
post-snapshot execution, 550
SCSI. *See* Small Computer Systems Interface
SDK. *See* Software Development Kit
Second normal form (2NF), 181
Secondary data file, 137
Secondary.mdf, 306
Secure Sockets Layer (SSL), 31–32 , 236 , 483
encryption, 580
support, 237–239
configuration, 238–239
Security, 460 . *See also* Data ; Data Transformation Service ; Databases ; Network ; Object ; Server

configuration, 262
enhancements, 21
ID (SID), 204 , 416
measures. *See* Intranet
mode, selection, 207–210
planning. *See* Structured Query Language Server
services, 5 , 6
simplification, 201
tab, 81–82
usage. *See* Cubes
SELECT
access, 632
clause, 664
icon, 672
permission, 222
query, 659 , 660 , 669
statement, 112 , 484 , 509 , 524
usage, 611–613 , 620–621 , 626 , 659
usage, 669
Select...For XML, 11–12 , 522–525
Self-joins, 619
Semantic analysis, 366
Semantic matching, 366
Semantic modeling format, 363
Semantic relationship, 367
Semi-synchronous mode, 591
Sequentially increasing fields, 662
Server, 301 . *See also* Advanced server ; Datacenter server ; Federated servers
assignation. *See* Distributor
availability. *See* Structured Query Language Server
cache, 496
configuration, 266 . *See also* Data ; Federated database servers
options, 48
settings, 132
creation. *See* Linked servers
FAQs, 240–242
hardware, upgrading, 585
instance support. *See* Multiple server instances
license, 62–63
monitoring, 266
network utility, 86–87
partitioning. *See* Data ; Federated database servers
performance, optimization, 673–687
publishing, enabling, 565–566
registration, 132 , 266
resources, management, 140

roles, 216–218
scalability. *See* Structured Query Language Server
security, 189 , 194–197
implementation, 231–239
settings tab, 83–84
synchronization, 564
Server/client access licensing model, 34
Serverwide operations, privileges assignation, 199
Service accounts, 141
configuration. *See* Structured Query Language Server 2000
Services, integration, 262
SESSIONPROPERTY function, 618
Set functions, 484
SET ROWCOUNT, 48
Settings | ANSI NULL default, 165
Settings | Auto close, 166
Settings | Auto shrink, 166
Settings | Auto update statistics, 165–166
Settings | Auto-create statistics, 166
Settings | Recursive triggers, 165
Settings | Torn page detection, 166
Settings | Use quoted identifiers, 166
s_Generation, 582
SGML. *See* Standard Generalized Markup Language
Shared dimensions, 463 , 480
Shared disks, 97 , 102
setup, 103
Shared session information, 19
Sharing, amount, 96
Shrinking operation, 302 , 337
SID. *See* Security
Simple backup-and-restore model, 303
Simple recovery, 165
Single-column integer values, 664
Site Server 3.0 Commerce Edition, 5
Small Computer Systems Interface (SCSI), 63 , 677
bus, 97
card, 110
channel, 99
smalldatatime (data type), 657
smallmoney (data type), 657
SMP. *See* Symmetric multiprocessor
SNA Server 4.0, 5
Snapshot
control. *See* ActiveX
folder, usage, 587

retrieval, FTP usage, 574–576
Snapshot Agent, 548 , 560 , 562–564
running, 587
Snapshot files
backup, 309
retention period, 310
Snapshot replication, 309 , 550 , 563–564
backup, 590
performance, enhancement process, 587–588
restoration, 590
Sniffer, 201
Snowflake schema, 481
Software
compatibility, 101
configuration, 678–683
RAID, 64
requirements. *See* Structured Query Language Server
Software Development Kit (SDK), 381 , 622
Source server, 115
Southwind
configuration, reviewing, 167–168
database, 158 , 184 , 186
Southwind Properties | Data Files, 162–163
Southwind Properties | Filegroups, 161–162
Southwind Properties | Options, 164–167
Southwind Properties | Permissions, 167
Southwind Properties | Transaction Log, 164
sp_ActiveDirectory_SCP, 249
Sp_configure, 686
sp_configure system-stored procedure, 47
sp_processmail, 291
sp_xml_preparedocument, 12 , 536–539
sp_xml_removedocument, 12 , 536–539
SQL. *See* Structured Query Language
SQL SERVER GROUP, 268
sql:after element, 535
SQL-DMF. *See* Structured Query Language Distributed Management Framework
SQL-DMO. *See* Structured Query Language Distributed Management Object
SQL-SCM API, 48
Sql_variant (data type), 601–603
SQRT, 618
s_RowLineage, 582
SSL. *See* Secure Sockets Layer
Standard Generalized Markup Language (SGML), 501
Standard logins, 198–199
Standard roles, 200 , 227–229

Star join, 470
Star schema, 470 , 481
Static IP addresses, 102
Statistics, updating, 173 , 336
STDEV statistical functions, 619
Storage. *See* Offline storage ; Offsite storage ; Online storage
locations, 459
media. *See* Backup
requirements. *See* Variable-length column storage requirements
determination, 307–311
system, 281
types. *See* Field
Storage area network (SAN), 65
Storage Design Wizard, 644
Stored procedures, 131 , 593 , 613 . *See also* Transact-SQL
number, 218
usage, 201
String data type, 529
String functions, 484
Striping, 146
Structured Query Language Distributed Management Framework (SQL-DMF), 639
Structured Query Language Distributed Management Object (SQL-DMO), 288–290 , 409 , 552 , 639–640
BulkCopy, 440–444
object models, 48
Structured Query Language (SQL). *See* Transact-SQL
2000 fail-over clustering architecture, 99
Agent job, 66 , 348
Analysis Services, 273
authentication mode, understanding, 205–206
database
management, 266
queries, 361
Debugger, 607–609
Enterprise Manager, DTS package execution, 430
execution, HTTP usage, 513–515
fail-over clustering, upgrading, 105–108
instance installation, 392
login, adding. *See* Enterprise Manager
Mail, 66 , 90–91 , 290–292
passwords, transmission, 206
Profiler, 665–673
trace file, loading. *See* Index Tuning Wizard
usage. *See* Northwind database
Project Wizard, 362 , 363
Query Analyzer, 669–673

replication services, 249–260
server objects, examination tools/techniques. *See* Active Directory
system administrator, 74
usage. *See* English Query
usage, timing. *See* ANSI SQL
XML view mapper, 530–534
downloading, 530–534
Structured Query Language (SQL) Server, 499
administration, 23–24
tools/techniques, 262–274
advanced installation, 75–76
Agent, 89–90 , 548
analysis services, 453
authentication, 207–210
availability, 93
backup, 295
backward compatibility, 47–48
CE edition, replication features, 580–582
clustering, planning, 99–102
data mining, 485–491
Database Designer, 184–186
databases
backup, 322–344
creation, 127 , 144–150
design, 127
moving/copying, 275–282
restoration, 344–354
DBA, 232
Developer Edition, 2 , 38
development tools/technologies, 14–20
DTS package, saving, 427
Enterprise Edition, 2
Enterprise Manager, 266–273
error log, 81
fail-over clustering, 98–109
FAQs, 125–126 , 188 , 356–358 , 498 , 649–650
identity values, assignation, 555
installation, 71–76 , 262
computer/processor requirements, 60
hard disk requirements, 61
hardware requirements, 60–61
memory requirements, 61
options, 63–71
planning, 59–71
requirements, 60–63
software requirements, 61–62

licensing, 34–35 , 62–63
limitations, overview, 518–519
login access, securing, 198–199
management, 93 ; 97
migration
planning, 44–46
steps, 46
MMC snap-ins, 273–274
objects, examination tools/techniques. *See* Active Directory
Personal Edition, 2 , 37–38
hardware requirements, 37–38
operating system, compatibility, 38
software requirements, 38
pricing, 34–35
program group, 30
programming
overview, 598–599
tools/technologies, 597
properties, 76–81 , 247–249
querying, HTTP usage, 512–518
recovery, 295
registration. *See* Active Directory
replication
architecture, 546–550
configuration, 565–579
features, 549–550
methods, 560–565
scalability, 93
security
management, 266
options, 202–231
planning, 190–202
understanding, 191–194
service accounts, creation, 65–66
services, 139–143
settings, 678–682
Standard Edition, 2
standard installation, 71–75
starting/stopping, 266
steps, success, 43–52
support, overview, 500–510
system databases, 131–135
unattended installation, 76
usage, 238–239
users, management, 266
version 6 . 5

fail-over clustering, 107–108
upgrading, 48–49
version 7 .0, 5 , 47
fail-over clustering, 106
upgrading, 49–52
versions, 550–551
XML support/limitations, 509
XPath limitations, 519
Structured Query Language (SQL) Server 2000, 6 . *See also* Windows CE
administrators, training, 44
architecture, 128–143
availability, 24–27
benefits, 42–43
configuration, 57
options/settings, 76–88
data files, securing, 197
Developer Edition, 32
edition requirements, 33
Enterprise Edition, 32 , 35–36
capacity limits, 35
hardware requirements, 35
operating system compatibility, 36
executable files, securing, 197
fail-over clustering, 49 , 108–109
FAQs, 54–56 , 92
features, 8–41
replication support, 549
improvements, 8–32
installation, 57
system backup, 58–59
maintenance, 171–180
migration, 1 , 46–52
development, 44
questions, 41–43
scope, determination, 43–44
monitoring, 171–180
named instances, 47
overview, 3–8
Personal Edition, 32 , 191
personnel, support, 44
programming features, 599–623
replication, 250–257
requirements, 32–41
scalability, 24–27
service accounts, configuration, 194–196
Standard Edition, 32

systems, inventory, 44
test
migration, performing, 44
plan, development, 44
upgrade process, 48 , 69–71
versions, 32–41
Windows CE Edition, 32 , 39–41
XML updategrams support, downloading, 534–535
Structured Query Language (SQL) Server 2000 Desktop Engine (MSDE), 32 , 39–41
Structured Query Language (SQL) Server Agent
information, 133
services, 30 , 142
starting, 327
Subdirectory, creation. *See* Databases
Sub-elements, 502
Subnets, 256 . *See also* Internet Protocol
Subqueries, 619
Subscribers, 71 , 547 , 565 , 574–575 , 589 . *See also* Heterogeneous subscribers ; Queued updating ; Windows CE subscribers
addition, 569–570
conflict minimization, 554
strategy restoration. *See* Failed subscriber
upgrading, 51–52
Subscription, 548 . *See also* Pull subscriptions ; Push subscriptions
configuration, 575–576
creation, 262
transformation, 549
Subsystem. *See* Disks
SUM, 619
Support personnel, training, 45–46
Supported query methods, HTTP usage, 513
Sybase system 1 .x, 34
Symmetric multiprocessor (SMP), 95
Synchronous mode, 590–591 . *See also* Semi-synchronous mode
System
administrator. *See* Structured Query Language
password assignation, 266
architectures, 96–97
databases, 30
backup, 335–344
restoration, 348–352
System area network (SAN), 101
System-stored procedures, 47 , 215 , 586

T

Table (data type), 603–604
Table user-defined functions, 611
Tables, 130 , 183 . *See also* Dimension table ; Dimensional table ; Fact table ; HyperText Markup Language ; Member tables
copying. *See* Databases
design, 656–657
enabling. *See* Full-text search
files, 675
full-text index, creation. *See* Northwind database
indexes, 445
RowGuid, management, 554–559
scans, 138 , 660
word occurrences, 395
XML data mapping. *See* Databases
TABLOCK, 445
TABLOCK hint, specification, 445
Tape device, 311 , 324
Tape library unit (TLU), 314
Tape unit, 299–300
Tasks, 413–414 . *See also* Data
creation, 423–426
frequency. *See* Export ; Import task
skipping, 416
types, 423–426
TCO. *See* Total cost of ownership
TCP/IP, 101 , 236 , 572
configuration, 103
listening, 573
usage, 78
TCSEC. *See* Trusted Computer System Evaluation Criteria
TempDB database, 132–133 , 199 , 221 , 678
files, 675
text (data), 135
text (data type), 657
Text documents, 381
Third normal form (3NF), 181 , 457
Third-party clients, 491
Third-party providers, 461
Third-party vendors, 465
Threshold, initiation, 416
Time-limited OLTP data, 456
Timestamp, 554
timestamp (data), 656
timestamp (data type), 657
TLU. *See* Tape library unit
TOP, 619

Total cost of ownership (TCO), 98
Total processor time, 684
TPC. *See* Transaction Processing Council
Trace
file, loading. *See* Index Tuning Wizard
setup. *See* Northwind database
Transaction logs, 137–138 , 305–306 , 316 , 324
backups, 318 , 338–339
disk directory, specification, 177
plan, specification, 176
size/growth rate, 307
files, 675
growth, defining, 157
naming, 156–157
scheduling intervals, 307
Transaction Processing Council (TPC), 96 , 652
benchmarks, 97–98
TPC-C, 98
TPC-H, 98
Transaction Server, 637
Transactional replication, 309 , 550 , 560–561 , 565–566
backup, 590
log shipping, usage, 590–591
performance, enhancement process, 587–588
restoration, 590
Transactions, 298
starting point, 306
Transact-SQL (T-SQL), 151 , 321–322 , 624–633
command, 333
execution, 140
OPENXML statement, 536–539
statement, 159 , 160 , 230 , 275 . *See also* CHECKDB
usage, 555
stored procedures, 231
usage, 209 . *See also* Databases ; Fixed server role ; Windows logins
Transfer Tasks, 27
Transformations, 414–415
Triggers, 131
enhancements, 16–17 , 614–617
size, 173
Trigger-specific inserted table, 616
TRN, 340
Truncation, 302
Trusted Computer System Evaluation Criteria (TCSEC), 193
T-SQL. *See* Transact-SQL
Tuple functions, 484

Two-click deployment, 362

U

UDT. *See* User-defined data type
Uncommitted replication transaction, 306
Uniform Resource Locator (URL), 494 , 510–513
usage, 500 , 517
UNION, 655
deleting, 672
operator, 653
UNION ALL, 659
UNIONS, 619
UNIQUEIDENTIFIER
column data type, usage, 555
data type, 555
Universal, 551
Universal Resource Locator (URL), 261
Unused space, removal. *See* Databases
UPDATE
action, 131
query, 660
statement, 113–114 , 121 , 525 , 560 , 576
usage, 613 , 626–627 , 656
Updategrams, 534–535
support. *See* Structured Query Language Server 2000
understanding, 535
UPDATETEXT statement, 630 , 631
Updating. *See* Queued updating
UPS protection, 101
URL. *See* Uniform Resource Locator ; Universal Resource Locator
URL query support. *See* HyperText Transfer Protocol
Usage Analysis Wizard, 495–496 , 644
Usage-Based Optimization Wizard, 496 , 644
User-defined data type (UDT), 130 , 131
User-defined functions, 18 , 130 , 167 , 609–613 . *See also* Inline user-defined functions ; Scalar user-defined functions ; Table user-defined functions
usage, 201
User-defined roles, 225 , 227–231
User-definition function, 131
Users, 492 . *See also* Active users ; Databases ; Enterprise-level user accounts. *See* Guest user account
adding. *See* Enterprise Manager
changing, 66–67
authentication, 232–234
connections, 687

management, 140
databases, 30 , 335
grouping, 200–201
permissions, assignation, 222–224 , 235
questions, understanding, 366–367

V

Variable-length column storage requirements, 149
VARIANCE statistical functions, 619
VB. *See* Visual Basic
View mapper. *See* Structured Query Language
Views, 131 . *See also* Distributed partitioned views ; Distributed view ; eXtensible Markup Language ; Indexed views ; Partitioned view
creation, 659
referencing, 619
usage, 201
Virtual cube, 487
Virtual dimension, 481
Virtual names, 512
Virtual Private Network (VPN), 202 , 236 , 571
usage. *See* Replicating
Virtual server, 75 , 102
Visual Basic (VB), 209 , 376 , 409–413 , 579 , 599
DTS, programmatic package execution, 432–433
references, 642
sample, 636–637
script file, DTS package saving, 429
Visual C++, 376 , 599
Visual Studio 6 .0, 5
Visual Studio 6.0 integration, 362
Visual Studio.NET, 7
VPN. *See* Virtual Private Network

W

W3C. *See* World Wide Web Consortium
W3C.org, 510
WAN. *See* Wide Area Network
Web. *See* World Wide Web
Weekly (occurrence), 329
WHERE clause, 484 , 611 , 654 , 664
Wide Area Network (WAN), 254 , 255 , 408 , 483
link, 553
Windows 98 , 38
Windows 2000, 67 , 233–234

cluster service component, installation, 104–105
IPSec, usage, 239
MSCS setup, 104–105
Windows 2000 Active Directory, 262–265
integration, 24 , 244–261
Windows 2000 Advanced Server, 36 , 38
Windows 2000 Datacenter Server, 36 , 38
Windows 2000 Professional, 38
Windows 2000 Server, 6 , 36
Windows 2000 settings, 682–683
Windows authentication, 81 , 207–210
mode, understanding, 202–205
Windows CE
SQL Server 2000, 2
subscribers, 581–582
Windows DNA, future, 4–8
Windows logins
adding, 210–216 . *See also* Enterprise Manager
Transact-SQL, usage, 213–216
granting/revoking/denying, 211
Windows Millennium Edition, 38
Windows NT, 67
authentication, 279
fibers, usage, 80
Windows NT 4.0 Server, 36 , 38
Windows operating system files, 675
Windows paging files, 675
Windows-only authentication mode, 210
WITH MEMBER statement, 485
WITH SCHEMABINDING option, 122 , 612 , 613
Word documents, 381
Word occurrences. *See* Tables
Workflow. *See* Packages
restarting, 416
World Wide Web Consortium (W3C), 500 , 501 , 510
World Wide Web (WWW / Web)
Analysis Services, access, 494
browser, 483
support, 31–32
Web-based English Query
applications, implementation, 376–377
solution, creation, 379–380
XML resources, 509–510
Write-enabled dimension type, 22
WRITETEXT statement, 630 , 631
WWW. *See* World Wide Web

X

X.500, 246
XDR. *See* eXtensible Markup Language
XML. *See* eXtensible Markup Language
XML.org, 510
XPath. *See* eXtensible Markup Language
xp_sendmail, 291
XSL. *See* eXtensible Stylesheet Language

List of Figures

Chapter 1: SQL Server 2000 Overview and Migration Strategies

Figure 1.1: The DNA Architecture Model.
Figure 1.2: The .NET Framework Model.

Chapter 2: Installing and Configuring SQL Server 2000

Figure 2.1: The Advanced Options screen.
Figure 2.2: The Main Installation screen for SQL Server Standard Edition.
Figure 2.3: The Select Computer dialog box.
Figure 2.4: The Installation Selection screen.
Figure 2.5: The Installation Definition screen.
Figure 2.6: The Instance Name screen.
Figure 2.7: The Setup Type screen.
Figure 2.8: The Select Components screen.
Figure 2.9: The Services Accounts screen.
Figure 2.10: The Authentication Mode screen.
Figure 2.11: The Choose Licensing Mode screen.
Figure 2.12: The Advanced Options screen.
Figure 2.13: The SQL Server Properties General tab.
Figure 2.14: The Startup Parameters dialog box.
Figure 2.15: The SQL Server Network Utility screen.
Figure 2.16: The SQL Server Properties Memory tab.
Figure 2.17: The SQL Server Properties Processor tab.
Figure 2.18: The SQL Server Properties Security tab.
Figure 2.19: The SQL Server Properties Connections tab.
Figure 2.20: The SQL Server Properties Server Settings tab.
Figure 2.21: The SQL Server Properties Database Settings tab.
Figure 2.22: The SQL Server Properties Replication tab.
Figure 2.23: The SQL Server Network Utility General tab
Figure 2.24: The SQL Server Network Utility Network Libraries tab.
Figure 2.25: The SQL Server Client Network Utility General tab.
Figure 2.26: The SQL Server Client Network Utility Alias tab.
Figure 2.27: The SQL Server Client Network Utility DB-Library tab.
Figure 2.28: The SQL Server Client Network Utility Network Libraries tab.
Figure 2.29: The SQL Server Agent Properties General tab.
Figure 2.30: The SQL Server Agent Properties Advanced tab.
Figure 2.31: The SQL Server Agent Properties Alert System tab.
Figure 2.32: The SQL Server Agent Properties Job System tab.
Figure 2.33: The SQL Server Agent Properties Connection tab.
Figure 2.34: The SQL Mail Configuration screen.

Chapter 3: SQL Server Scalability and Availability

Figure 3.1: System architecture affects database scalability.
Figure 3.2: Planning your SQL Server 2000 Fail-Over Clustering environment.
Figure 3.3: Select the Cluster Service component.
Figure 3.4: Federated servers with partitioned data and links.
Figure 3.5: The three server roles in log shipping: source, destination, and monitor.
Figure 3.6: The Select Databases dialog box.
Figure 3.7: Adding or Editing the log-shipping destination database.
Figure 3.8: Setting the log-shipping schedules.
Figure 3.9: Specifying the log-shipping alert thresholds.
Figure 3.10: Specifying the log-shipping monitor server information.
Figure 3.11: Modifying the log-shipping source schedule.
Figure 3.12: Modifying the log-shipping destination schedule and other properties.

Chapter 4: Designing and Creating SQL Server Databases

Figure 4.1: SQL Server architecture overview.
Figure 4.2: A normalized database example.
Figure 4.3: The Rebuild Master Database utility.
Figure 4.4: Data storage architecture.
Figure 4.5: SQL Server Service Manager.
Figure 4.6: The SQL Server Properties (Configure) dialog box.
Figure 4.7: Northwind database File Autogrow Properties.
Figure 4.8: The Southwind logical database model.
Figure 4.9: The General tab of the Southwind Properties dialog box.
Figure 4.10: The Locate Database File dialog box.
Figure 4.11: The Edit Recurring Job Schedule dialog box.
Figure 4.12: The Database Maintenance Plan dialog box.
Figure 4.13: The Database Maintenance Plan History dialog box.
Figure 4.14: Southwind database entity-relationship diagram.
Figure 4.15: The Table Properties dialog box.

Chapter 5: Database and Server Security

Figure 5.1: The various levels of security in a SQL Server 2000 environment.
Figure 5.2: SQL Server 2000 service startup service account settings.
Figure 5.3: SQL Server file system security.
Figure 5.4: An overview of the SQL Server security model.
Figure 5.5: The Windows authentication login process.
Figure 5.6: SQL authentication login process.
Figure 5.7: Enabling mixed-mode authentication during installation.
Figure 5.8: Settings for mixed-mode security.

Figure 5.9: The Security tree in Enterprise Manager.
Figure 5.10: Adding a new Windows login.
Figure 5.11: Adding a new standard SQL login.
Figure 5.12: Fixed server role membership.
Figure 5.13: Northwind database users.
Figure 5.14: Database user properties.
Figure 5.15: Setting permissions in Enterprise Manager.
Figure 5.16: Changing a user's role membership.
Figure 5.17: Adding a new database role.
Figure 5.18: Creating a new application role.
Figure 5.19: A security scenario.
Figure 5.20: Server Network Utility with the multiprotocol library enabled.
Figure 5.21: Enabling Multiprotocol Encyrption.
Figure 5.22: Forcing protocol encryption.

Chapter 6: Administration and Active Directory Integration

Figure 6.1: The Active Directory dialog box in SQL Server Properties.
Figure 6.2: Successful update of Active Directory properties.
Figure 6.3: SQL Server information appearing in Active Directory Users and Computers.
Figure 6.4: Listing a database in Active Directory.
Figure 6.5: New database object shown in Active Directory.
Figure 6.6: The LDP tool.
Figure 6.7: The ADSI Edit tool.
Figure 6.8: Editing attributes in ADSI Edit.
Figure 6.9: SQL Server changes shown in the Active Directory Sites and Services console.
Figure 6.10: Listing a publication in Active Directory.
Figure 6.11: Looking into Active Directory.
Figure 6.12: Specify Publication.
Figure 6.13: Advanced search options for the Active Directory listed publications.
Figure 6.14: Registering Active Directory.
Figure 6.15: The Active Directory Users and Computers console.
Figure 6.16: The user creation dialog box.
Figure 6.17: Password information dialog box.
Figure 6.18: The User Properties dialog box.
Figure 6.19: Creating a new server group.
Figure 6.20: Login properties.
Figure 6.21: New user creation.
Figure 6.22: The Microsoft Full-Text Search Service properties dialog box.
Figure 6.23: SQL Server Agent Properties.
Figure 6.24: DTC Transaction Statistics in the Component Services MMC.
Figure 6.25: The Analysis Services MMC.
Figure 6.26: The Restore database dialog box.

Figure 6.27: The Management Wizards list.
Figure 6.28: The default destination server is <local>.
Figure 6.29: Select the Databases to Move or Copy.
Figure 6.30: Detaching a database.
Figure 6.31: Attaching a database.
Figure 6.32: Linked Server dialog.
Figure 6.33: Select databases to be maintained.
Figure 6.34: The Database Integrity Check dialog box.
Figure 6.35: Summary of the maintenance plan.
Figure 6.36: The SQL Mail Configuration dialog box.
Figure 6.37: Operator properties.

Chapter 7: SQL Server Backup and Recovery

Figure 7.1: Standard versus fail-safe server configurations.
Figure 7.2: Transaction log structure.
Figure 7.3: The welcome screen of the Create Database Backup Wizard.
Figure 7.4: Backup jobs available through the SQL Server Enterprise Manager.
Figure 7.5: The welcome screen of the Database Maintenance Plan Wizard.
Figure 7.6: Maintenance plans available through the SQL Server Enterprise Manager.
Figure 7.7: Wizard selection.
Figure 7.8: The Select Database to Backup dialog box.
Figure 7.9: Select Backup Destination and Action dialog box.
Figure 7.10: The Backup Verification and Scheduling dialog box.
Figure 7.11: The Edit Schedule dialog box.
Figure 7.12: The Edit Recurring Job Schedule dialog box.
Figure 7.13: Completing the Create Database Backup Wizard.
Figure 7.14: A summary of weekly differential backups.
Figure 7.15: The Job Properties dialog box.
Figure 7.16: The Edit Job Step dialog box.
Figure 7.17: The Select Databases dialog box.
Figure 7.18: The Update Data Optimization Information dialog box.
Figure 7.19: The Database Integrity Check dialog box.
Figure 7.20: Specify the Transaction Log Backup Plan dialog box.
Figure 7.21: The Specify Transaction Log Backup Disk Directory dialog box.
Figure 7.22: The Maintenance History dialog box.
Figure 7.23: The Restore Database dialog box.
Figure 7.24: Restore filegroups or files.
Figure 7.25: Restore from device.
Figure 7.26: The Northwind Properties Options tab.
Figure 7.27: SQL Server Properties Database Settings tab.

Chapter 8: Microsoft English Query and Full-Text Search

Figure 8.1: English Query run-time architecture.
Figure 8.2: The Install Components screen in the SQL Server 2000 Setup application.
Figure 8.3: English Query Setup.
Figure 8.4: The Project Explorer view showing the English Query Project Module and Regression files.
Figure 8.5: The OLE DB Provider Properties tab.
Figure 8.6: The Data Link Connection Properties tab.
Figure 8.7: The New Project Dialog box.
Figure 8.8: The English Query Project Wizard.
Figure 8.9: The Entity dialog box for customer.
Figure 8.10: The Customers_have_employees Relationship dialog box.
Figure 8.11: Testing an English Query Model with the Model Test application.
Figure 8.12: Regression text XML in the English Query editor.
Figure 8.13: A functioning Web site that can interact with the user via natural language.
Figure 8.14: The full-text search architecture.
Figure 8.15: The Full-Text Catalog Properties dialog box.
Figure 8.16: Full-Text Indexing Schedules for a catalog.
Figure 8.17: The New Full-Text Indexing Catalog Schedule dialog box.
Figure 8.18: The SQL Server 2000 English Query full-text architecture.
Figure 8.19: A potential disk configuration.

Chapter 9: Importing and Exporting Data

Figure 9.1: DTS architectural overview.
Figure 9.2: The Package Properties dialog box.
Figure 9.3: DTS Export Wizard data source selection.
Figure 9.4: The DTS Designer.
Figure 9.5: The ActiveX Scripts task.
Figure 9.6: The Save DTS Package dialog box.
Figure 9.7: The dtsrunui.exe user interface.

Chapter 10: SQL Server Analysis Services

Figure 10.1: The dimension model illustrating a star schema.
Figure 10.2: The Analysis Services architecture.
Figure 10.3: Analysis Manager, showing an expanded tree.
Figure 10.4: Working with the Sales cube in the Analysis Services Processing Task dialog box.
Figure 10.5: Selecting a fact table with the Cube Wizard.
Figure 10.6: Selecting measures with the Cube Wizard.
Figure 10.7: Selecting dimensions with the Cube Wizard.
Figure 10.8: The results of the Wizard: The Cube Editor.
Figure 10.9: Defining the parent-child hierarchy in the Dimension Wizard.
Figure 10.10: Results obtained using the OLAP Mining Model Editor, showing numerical attributes.

Figure 10.11: The browser showing attributes graphically.
Figure 10.12: The mining browser, showing results from the Relational Mining Model Editor.
Figure 10.13: The Relational Mining Model Editor.

Chapter 11: Using XML with SQL Server

Figure 11.1: XML tree structure.
Figure 11.2: A partitioned XML tree structure.
Figure 11.3: Virtual directory name and path settings.
Figure 11.4: Adding a virtual name.
Figure 11.5: The result set with a ROOT element error.
Figure 11.6: A new project in the SQL Server View Mapper.
Figure 11.7: Data Link Properties.
Figure 11.8: A SQL Server XML View Mapper project.
Figure 11.9: Using the XPath Query tool.
Figure 11.10: Executing the OPENXML function.
Figure 11.11: Verifying that the record was inserted.

Chapter 12: Database Replication Techniques and Configuration

Figure 12.1: SQL Server 2000 replication roles.
Figure 12.2: Transactional replication.
Figure 12.3: Merge replication.
Figure 12.4: Snapshot replication.
Figure 12.5: Configuring replication in Enterprise Manager.
Figure 12.6: The Select Distributor screen.
Figure 12.7: Choosing updateable subscription options.
Figure 12.8: Setting the system to allow anonymous subscriptions.
Figure 12.9: Setting up an FTP server as a snapshot location.
Figure 12.10: The Conflict Viewer.
Figure 12.11: The Set Subscription Priority screen in the Pull Subscription Wizard.
Figure 12.12: Windows CE replication.
Figure 12.13: Setting up publication to Windows CE clients.
Figure 12.14: Adding the server to Active Directory.
Figure 12.15: Adding an existing Publication to Active Directory.
Figure 12.16: The Specify Publication screen in the Pull Subscription Wizard.
Figure 12.17: Browsing for a publication in Active Directory.

Chapter 13: Programming Tools and Technologies in SQL Server

Figure 13.1: Query Analyzer Object Browser and templates.

Figure 13.2: Execute Procedure parameter entry.
Figure 13.3: The SQL Debugger.
Figure 13.4: ADO, OLE DB, and ODBC.
Figure 13.5: The DMO object model.
Figure 13.6: The DTS object hierarchy.
Figure 13.7: The SQL-DMO Replication object model.

Chapter 14: Performance-Tuning Tools and Techniques

Figure 14.1: An example of a federated database.
Figure 14.2: Communication among linked servers.
Figure 14.3: Classes of indexes.
Figure 14.4: SQL Profiler with a trace running.
Figure 14.5: The Trace Properties dialog box General tab.
Figure 14.6: The Trace Properties dialog box Filters tab.
Figure 14.7: An execution plan in Query Analyzer.
Figure 14.8: The next step of our execution plan in Query Analyzer.
Figure 14.9: SQL Server Properties.
Figure 14.10: The Microsoft Management Console snap-in.
Figure 14.11: Data structures in the Northwind database that could become part of a distributed partitioned view.
Figure 14.12: A database server with RAID5, RAID1, and RAID0 arrays.

List of Tables

Chapter 1: SQL Server 2000 Overview and Migration Strategies

Table 1.1: Microsoft SQL Server 2000 License Estimated Retail Pricing
Table 1.2: SQL Server 2000 Edition Features Comparison
Table 1.3: SQL Server 2000 Recovery Models

Chapter 2: Installing and Configuring SQL Server 2000

Table 2.1: SQL Server 2000 Software Requirements
Table 2.2: Startup Options Available for SQL Server 2000

Chapter 4: Designing and Creating SQL Server Databases

Table 4.1: SQL Server Maximum Capacities
Table 4.2: RAID Levels and Performance
Table 4.3: Southwind Logical Database Properties
Table 4.4: Southwind Physical Database Properties
Table 4.5: An Unnormalized City Table
Table 4.6: A Revised City Table
Table 4.7: A Founder Table
Table 4.8: A Suburb Table
Table 4.9: The 2NF City Table
Table 4.10: The 2NF State Table
Table 4.11: The 2NF Founder Table
Table 4.12: The 2NF Suburb Table
Table 4.13: The 3NF City Table

Chapter 5: Database and Server Security

Table 5.1: Default SQL Server Logins
Table 5.2: Transact-SQL Statements for Granting, Revoking, and Denying Windows Logins
Table 5.3: Transact-SQL Statements for Granting, Revoking, and Denying Windows Logins
Table 5.4: Fixed Server Roles
Table 5.5: Transact-SQL Statements for Managing Fixed Server Roles
Table 5.6: Transact-SQL Statements for Granting and Revoking Database Access
Table 5.7: Assignable Permissions

Table 5.8: Transact-SQL Statements for Granting, Revoking, and Denying Permissions
Table 5.9: Fixed Database Roles
Table 5.10: Transact-SQL Statements for Managing Database Roles
Table 5.11: Transact-SQL Statements for Creating and Deleting Standard Database Roles
Table 5.12: Transact-SQL Statements for Creating, Deleting, and Activating Application Roles

Chapter 6: Administration and Active Directory Integration

Table 6.1: SQL Server Agent Options
Table 6.2: DBCC Statements

Chapter 7: SQL Server Backup and Recovery

Table 7.1: Capacities of Various Tape Technologies
Table 7.2: Media-Related Terms
Table 7.3: Example File and Drive Assignments for the Northwind Backup Scheme
Table 7.4: Explanation of the BACKUP Commands
Table 7.5: The sysmaint Extended Stored Procedure and Utility Parameters
Table 7.6: RESTORE Command Arguments

Chapter 8: Microsoft English Query and Full-Text Search

Table 8.1: Common RAID Levels

Chapter 9: Importing and Exporting Data

Table 9.1: Tasks That Copy and Manage Data
Table 9.2: Tasks That Transform Data
Table 9.3: Tasks That Execute Jobs
Table 9.4: Arguments Used with dtsrun.exe
Table 9.5: Data Type Conversions for Format Files
Table 9.6: Common Delimiters
Table 9.7: Data File Type Property Enumeration Values
Table 9.8: Arguments Used with BCP
Table 9.9: Export Method Comparisons

Chapter 10: SQL Server Analysis Services

Table 10.1: Analysis Services Components and Default Locations

Chapter 11: Using XML with SQL Server

Table 11.1: XML Reserved Characters
Table 11.2: XPath Axes and Identifiers
Table 11.3: XPath Functions
Table 11.4: URL Query Arguments
Table 11.5: XML Template Tags
Table 11.6: XPath Functions Implemented in SQL Server 2000
Table 11.7: XPath Functions Not Implemented in SQL Server 2000
Table 11.8: XPath to XDR to SQL Server Data Type Conversions
Table 11.9: FOR XML Syntax Arguments
Table 11.10: XDR Schema Language Annotations
Table 11.11: Data Type Conversions
Table 11.12: Mapping Between SQL Server and XML Data Types
Table 11.13: OPENXML Function Arguments
Table 11.14: sp_xml_preparedocument Stored Procedure Arguments
Table 11.15: New OLE DB Properties for XML in SQL Server 2000

Chapter 13: Programming Tools and Technologies in SQL Server

Table 13.1: Data Type Family Groupings

Chapter 14: Performance-Tuning Tools and Techniques

Table 14.1: Guidelines for Partitioning
Table 14.2: Common RAID Levels
Table 14.3: Ordinary Configuration Options
Table 14.4: Advanced Configuration Options

List of Sidebars

Chapter 1: SQL Server 2000 Overview and Migration Strategies

Is .NET the End of DNA?

Chapter 2: Installing and Configuring SQL Server 2000

Backing Up a System Prior to Installing SQL Server 2000

Chapter 3: SQL Server Scalability and Availability

System Architectures and Database Scalability
How to Recover from Fail-Over Cluster Failure

Chapter 4: Designing and Creating SQL Server Databases

Database Normalization

Chapter 5: Database and Server Security

Manage By Groups
Using Transact SQL Instead of Enterprise Manager

Chapter 6: Administration and Active Directory Integration

About Active Directory
Tools and Techniques for Examining SQL Server Objects in Active Directory
Active Directory Replication
The Delegation of Administration Wizard
Managing a SQL Server

Chapter 7: SQL Server Backup and Recovery

Backup and Recovery Strategies
Backup-and-Restore Strategies in Replication and Publish-and-Subscribe Environments

Chapter 9: Importing and Exporting Data

A Tool for Any Environment
Reusing Existing Packages

Chapter 10: SQL Server Analysis Services

Unattended Installations

Chapter 12: Database Replication Techniques and Configuration

Setting the NOT FOR REPLICATION Flag on Check Constraints
When to Perform Additional Backups

Chapter 13: Programming Tools and Technologies in SQL Server

When to Use ANSI SQL

Chapter 14: Performance-Tuning Tools and Techniques

Sp_configure