LOVELY PROFESSIONAL UNIVERSITY

DATABASE ADMINISTRATION

Edited By

Sartaj Singh

Printed by
EXCEL BOOKS PRIVATE LIMITED
A-45, Naraina, Phase-I,
New Delhi-110028
for
Lovely Professional University
Phagwara

SYLLABUS

Database Administration

Objectives: To Impart the skills needed to administer database of an organization using SQL Server. Student will learn: how to create and manage data of an organization, providing authorization and authentication, network communication models, automatic administrative tasks, backup and recovery models, SQL server performance issues and integration services.

Sr. No.	Description
1.	***SQL Server RDBMS:*** What is SQL server, Different Editions of SQL Server, Architecture and Database Objects, SQL server databases, Database storage, Security: Windows and SQL Server Authentication
2.	***Installing SQL Server:*** Installation Planning, requirements ***SQL Server Tools:*** Overview of Management Studio , Log File Viewer
3.	***SQL Server Storage Architecture:*** Resource database, Database physical structure, Database Files, Transaction Log
4.	***SQL Server Databases:*** System Databases. User Databases. Database Planning, Creating Databases, Tables, Constraints, Database Diagrams, Views, Synonyms, Programming Objects – Functions, Procedures, Triggers
5.	***SQL Server Authentication Modes:*** Principals, Permissions, SQL Server Encryption Overview
6.	***Configuring SQL Server Network Communication:*** Network Protocols, Native Client Configuration
7.	***Automating Administrative Tasks:*** Database Mail, Event Notifications, SQL Server Agent, Maintenance Plans.
8.	***Disaster Prevention and Recovery:*** Database Recovery Models. **Database Backup:** Back Types and options, Backup Strategies. Restoring Databases. Database Snapshots
9.	***Monitoring SQL Server for Performance:*** Overview, Tools and Techniques for Monitoring Performance, Monitoring Database Modifications
10.	***SQL Server Integration Services:*** How to import and export data into and from SQL server.

CONTENT

Unit 1: SQL Server RDBMS 1
Sartaj Singh, Lovely Professional University

Unit 2: Installing SQL Server 12
Sartaj Singh, Lovely Professional University

Unit 3: SQL Server Tools 38
Sartaj Singh, Lovely Professional University

Unit 4: SQL Server Storage Architecture 44
Sartaj Singh, Lovely Professional University

Unit 5: SQL Server Databases 55
Sartaj Singh, Lovely Professional University

Unit 6: SQL Server Authentication 84
Sartaj Singh, Lovely Professional University

Unit 7: Automating Administrating Tasks 98
Deepak Mehta, Lovely Professional University

Unit 8; Configuring SQL Server Network Communication 114
Deepak Mehta, Lovely Professional University

Unit 9: Database Recovery Models 122
Deepak Mehta, Lovely Professional University

Unit 10: Database Backup and Restore 131
Deepak Mehta, Lovely Professional University

Unit 11: Monitoring SQL Server for Performance 155
Sartaj Singh, Lovely Professional University

Unit 12: Tools and Techniques for Monitoring Performance 163
Sartaj Singh, Lovely Professional University

Unit 13: Monitoring Database Modifications 179
Deepak Mehta, Lovely Professional University

Unit 14: SQL Server Integration Services 188
Deepak Mehta, Lovely Professional University

Unit 1: SQL Server RDBMS

Notes

CONTENTS

Objectives
Introduction
1.1 What is SQL?
1.2 What is SQL SERVER?
1.3 Different Editions of SQL Server
1.4 Architecture and Database Objects
1.4.1 Database
1.4.2 Relational Database
1.4.3 Client/Server
1.4.4 Structured Query Language (SQL)
1.5 SQL Server Databases
1.6 Database Storage
1.7 Security: Windows and SQL Server Authentication
1.8 Summary
1.9 Keywords
1.10 Review Questions
1.11 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand the concept of SQL and SQL Server
- Know about different editions of SQL Server
- Describe the basic architecture of SQL server
- Learn about the security features of SQL Server

Introduction

The SQL language and relational database systems based on it are one of the most important foundation technologies in the computer industry. Over the last two decades, SQL has grown from its first commercial use into a computer product and services market segment worth tens of billions of dollars per year, and SQL stands today as the standard computer database language. Literally hundreds of database products now support SQL, running on computer systems from mainframes to personal computers and even handheld devices. Virtually every major enterprise software product relies on SQL for its data management, and SQL is at the core of the database products from Microsoft, Oracle, and IBM, the three largest software companies in the world. SQL is also at the heart of open-source database products that are helping to fuel the popularity of Linux and the open-source movement. From its obscure beginnings as an IBM research project, SQL has leaped to prominence as both an important computer technology and a powerful market force.

Notes

1.1 What is SQL?

SQL is a tool for organizing, managing, and retrieving data stored by a computer database. The acronym SQL is an abbreviation for Structured Query Language. For historical reasons, SQL is usually pronounced "sequel," but the alternate pronunciation "S.Q.L." is also used. As the name implies, SQL is a computer language that you use to interact with a database. In fact, SQL works with one specific type of database, called a relational database.

Figure 1.1: Use of SQL for Database Access

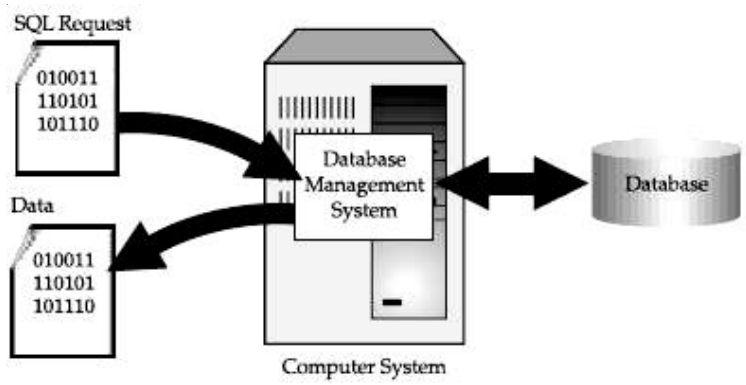


Figure 1.1 shows how SQL works. The computer system in the figure has a database that stores important information. If the computer system is in a business, the database might store inventory, production, sales, or payroll data. On a personal computer, the database might store data about the checks you have written, lists of people and their phone numbers, or data extracted from a larger computer system. The computer program that controls the database is called a database management system, or DBMS.

When you need to retrieve data from a database, you use the SQL language to make the request. The DBMS processes the SQL request, retrieves the requested data, and returns it to you. This process of requesting data from a database and receiving back the results is called a database query—hence the name Structured Query Language.

Notes The name Structured Query Language is actually somewhat of a misnomer. First of all, SQL is far more than a query tool, although that was its original purpose, and retrieving data is still one of its most important functions.

SQL is used to control all of the functions that a DBMS provides for its users, including:

1. ***Data definition:*** SQL lets a user define the structure and organization of the stored data and relationships among the stored data items.
2. ***Data retrieval:*** SQL allows a user or an application program to retrieve stored data from the database and use it.
3. ***Data manipulation:*** SQL allows a user or an application program to update the database by adding new data, removing old data, and modifying previously stored data.
4. ***Access control:*** SQL can be used to restrict a user's ability to retrieve, add, and modify data, protecting stored data against unauthorized access.

Notes

5. ***Data sharing:*** SQL is used to coordinate data sharing by concurrent users, ensuring that they do not interfere with one another.

6. ***Data integrity:*** SQL defines integrity constraints in the database, protecting it from corruption due to inconsistent updates or system failures.

SQL is thus a comprehensive language for controlling and interacting with a database management system.

Second, SQL is not really a complete computer language like COBOL, C, C++, or Java. SQL contains no IF statement for testing conditions, and no GOTO, DO, or FOR statements for program flow control. Instead, SQL is a database sub-language, consisting of about 40 statements specialized for database management tasks. These SQL statements can be embedded into another language, such as COBOL or C, to extend that language for use in database access. Alternatively, they can be explicitly sent to a database management system for processing, via a call-level interface from a language such as C, C++, or Java, or via messages sent over a computer network.

Finally, SQL is not a particularly structured language, especially when compared to highly structured languages such as C, Pascal, or Java. Instead, SQL statements resemble English sentences, complete with "noise words" that don't add to the meaning of the statement but make it read more naturally. There are quite a few inconsistencies in the SQL language, and there are also some special rules to prevent you from constructing SQL statements that look perfectly legal, but don't make sense.

Did u know? Despite the inaccuracy of its name, SQL has emerged as the standard language for using relational databases. SQL is both a powerful language and one that is relatively easy to learn.

1.2 What is SQL Server?

Microsoft SQL Server is a database management and analysis system for e-commerce, line-of-business, and data warehousing solutions. In this section you will find information for several versions of SQL Server. You will find articles on database and database application design, as well as examples of the uses that SQL Server can be put to.

SQL Server 2008, the latest version, includes enhanced XML support, integration of .NET Framework objects in databases, improved integration with Microsoft Visual Studio and the Microsoft Office System, as well as improved analysis, reporting, and data integration services.

1.3 Different Editions of SQL Server

Microsoft makes SQL Server available in multiple editions, with different feature sets and targeting different users. These editions are:

Mainstream Editions

Datacenter

SQL Server 2008 R2 Datacenter is the full-featured edition of SQL Server and is designed for datacenters that need the high levels of application support and scalability. It supports 256 logical processors and virtually unlimited memory. Comes with StreamInsight Premium edition.

Notes

Enterprise

Enterprise is the high-end edition with the full attribute set. SQL Server 2008 R2 Enterprise provides a complete, trusted data platform for challenging, mission-critical applications, BI solutions, and reporting. Some of the new traits incorporated in this edition comprise assistance for up to eight processors, conscription of up to 25 managed instances of SQL Server into a single Utility Control Point, Power Pivot for SharePoint, data density support for UCS-2 Uni-code, Master Data Services, assistance for up to four virtual machines, and the potential to maintain up to 2 terabytes of RAM. It still offers high levels of accessibility, scalability, and safety, and provides classic SQL Server 2008 traits like data and backup compression, Resource Governor, Transparent Data Encryption (TDE), advanced data mining algorithms, mirrored backups, and Oracle publishing.

SQL Server Enterprise Edition includes both the core database engine and add-on services, with a range of tools for creating and managing a SQL Server cluster. It can manage databases as large as 524 petabytes and address 2 terabytes of memory and supports 8 physical processors.

Standard

SQL Server 2008 R2 Standard is an inclusive data management and BI platform that offers medium-class solutions for minor organizations. It does not comprise all the bells and whistles incorporated in Datacenter and Enterprise; though, it continues to provide best-in-class ease of use and manageability. Backup compression, which was an enterprise trait with SQL Server 2008, is now a trait included with the SQL Server 2008 R2 Standard. Compared to Datacenter and Enterprise, Standard assists only up to four processors, up to 64 GB of RAM, one virtual machine, and two failover clustering nodes

SQL Server Standard edition includes the core database engine, along with the stand-alone services. It differs from Enterprise edition in that it supports fewer active instances (number of nodes in a cluster) and does not include some high-availability functions such as hot-add memory (allowing memory to be added while the server is still running), and parallel indexes.

Web

At a much more reasonable price compared to Datacenter, Enterprise, and Standard, SQL Server 2008 R2 Web is concentrated on service suppliers hosting Internet-facing Web serving surroundings. Dissimilar to Workgroup and Express, this edition doesn't have a tiny database size restriction, and it assists four processors and up to 64 GB of memory. SQL Server 2008 R2 Web does not provide the similar premium traits located in Datacenter, Enterprise, and Standard; though, it is still the ideal platform for hosting Websites and Web applications.

SQL Server Web Edition is a low-TCO option for Web hosting.

Workgroup

Workgroup is a cost-effective, protected, and dependable database and reporting platform meant for implementing smaller workloads than Standard. For instance, this edition is perfect for branch office solutions like branch data storage, branch reporting, and remote synchronization. Comparable to Web, it assists a maximum database size of 524 terabytes; though, it assists only two processors and up to 4 GB of RAM. It is possible to improve Workgroup to Standard or Enterprise.

SQL Server Workgroup Edition includes the core database functionality but does not include the additional services.

Notes

Express

SQL Server Express Edition is a scaled down, free edition of SQL Server, which includes the core database engine. While there are no limitations on the number of databases or users supported, it is limited to using one processor, 1 GB memory and 4 GB database files (10 GB database files from SQL Server Express 2008 R2). The entire database is stored in a single .mdf file, and thus making it suitable for XCOPY deployment. It is intended as a replacement for MSDE. Two additional editions provide a superset of features not in the original Express Edition. The first is **SQL Server Express with Tools**, which includes SQL Server Management Studio Basic. **SQL Server Express with Advanced Services** adds full-text search capability and reporting services.

Specialized Editions

Azure

SQL Azure offers many of the similar means of SQL Server distributed as a cloud service on the Windows Azure Platform. Dissimilar to other editions of SQL Server, you do not require to prerequisite hardware for, install or patch SQL Azure; Microsoft sustains the platform for you. You have to set up a subscription, prerequisite your service, and begin using SQL Azure database. You also do not require architecting a database installation for scalability, high accessibility or disaster revival as these traits are offered mechanically by the service. Any application that accesses SQL Azure must have Internet access so as to connect to the database.

Preferably, applications that utilize SQL Azure Database are constructed for and deployed to the Windows Azure Platform. This offers the highest performance and dependability for database connections since all of the communication among the application tier and the data tier occur within Microsoft data centers and utilize private high speed networks. You can arrange SQL Azure to sustain connections from applications running on-premise, but the eminence of the connection will be restricted to the bandwidth and consistency of your internet connection.

Microsoft SQL Azure Database is the cloud-based version of Microsoft SQL Server, presented as software as a service on Azure Services Platform.

Compact (SQL CE)

The compact edition is an embedded database engine. Unlike the other editions of SQL Server, the SQL CE engine is based on SQL Mobile (initially designed for use with hand-held devices) and does not share the same binaries. Due to its small size (1 MB DLL footprint), it has a markedly reduced feature set compared to the other editions. For example, it supports a subset of the standard data types, does not support stored procedures or Views or multiple-statement batches (among other limitations). It is limited to 4 GB maximum database size and cannot be run as a Windows service, Compact Edition must be hosted by the application using it. The 3.5 version includes supports ADO.NET Synchronization Services. SQL CE does not support ODBC connectivity, unlike SQL Server proper.

Developer

SQL Server Developer Edition includes the same features as SQL Server Datacenter Edition, but is limited by the license to be only used as a development and test system, and not as production server. This edition is available to download by students free of charge as a part of Microsoft's DreamSpark program.

Notes

Embedded (SSEE)

Windows Internal Database is an alternative of SQL Server Express 2005 that is incorporated with Windows Server 2008 and Windows Server 2008 R2, and is integrated with other free Microsoft products released after 2007 that need an SQL Server database backend. Windows SharePoint Services 3.0 and Windows 3.0 both comprise Windows Internal Database, which can be accessed as an alternative to by means of a retail edition of SQL Server. WID is a 32-bit application, even as constituent of Windows Server 2008 64-bit, which installs in the path C:\Program Files (x86)\Microsoft SQL Server. Windows Internal Database is not obtainable as a separate product for use by end-user applications; Microsoft offers SQL Server Express and Microsoft SQL Server for this purpose. In addition, it is intended to only be accessible to Windows Services running on the similar machine.

SQL Server 2005 Embedded Edition is a specially configured named instance of the SQL Server Express database engine which can be accessed only by certain Windows Services.

Evaluation

SQL Server Evaluation Edition, also known as the *Trial Edition*, has all the features of the Enterprise Edition, but is limited to 180 days, after which the tools will continue to run, but the server services will stop.

Fast Track

SQL Server Fast Track Data Warehouse is a new set of reference architectures used for data warehousing that will assist eradicate many of the hurdles companies usually face while making data warehouses. For as small as $13,000 per terabyte, SQL Server Fast Track Data Warehouse provides customers instant access to pretested, preconfigured industry-standard hardware from partners Bull, Dell and HP that augments Microsoft SQL Server 2008 scalability up to 32 terabytes, and helps decrease the time and attempt required to deploy mission-critical projects.

SQL Server Fast Track is specifically for enterprise-scale data warehousing storage and business intelligence processing, and runs on reference-architecture hardware that is optimized for Fast Track.

Parallel Data Warehouse (PDW)

A massively parallel processing (MPP) SQL Server appliance optimized for large-scale data warehousing such as hundreds of terabytes.

Self Assessment

Name the following:

1. A database management and analysis system for e-commerce, line-of-business, and data warehousing solutions.
2. A low-TCO option for Web hosting.
3. The cloud-based version of Microsoft SQL Server.
4. The full-featured edition of SQL Server and is designed for datacenters that need the high levels of application support and scalability.
5. The SQL Server edition that includes both the core database engine and add-on services, with a range of tools for creating and managing a SQL Server cluster.

Notes

1.4 Architecture and Database Objects

Microsoft SQL Server is a Structured Query Language (SQL) based, client/server relational database. Each of these terms describes a fundamental part of the architecture of SQL Server.

1.4.1 Database

A database is similar to a data file in that it is a storage place for data. Like a data file, a database does not present information directly to a user; the user runs an application that accesses data from the database and presents it to the user in an understandable format.

Database systems are more powerful than data files. The data is more highly organized. In a well-designed database, there are no duplicate pieces of data that the user or application has to update at the same time. Related pieces of data are grouped together in a single structure or record, and relationships can be defined between these structures and records.

When working with data files, an application must be coded to work with the specific structure of each data file. In contrast, a database contains a catalog that applications use to determine how data is organized. Generic database applications can use the catalog to present users with data from different databases dynamically, without being tied to a specific data format.

A database typically has two components: the files holding the physical database and the database management system (DBMS) software that applications use to access data. The DBMS is responsible for enforcing the database structure, including:

- Maintaining the relationships between data in the database.
- Ensuring that data is stored correctly, and that the rules defining data relationships are not violated.
- Recovering all data to a point of known consistency in case of system failures.

1.4.2 Relational Database

There are different ways to organize data in a database but relational databases are one of the most effective. Relational database systems are an application of mathematical set theory to the problem of effectively organizing data. In a relational database, data is collected into tables (called relations in relational theory).

A table represents some class of objects that are important to an organization. For example, a company may have a database with a table for employees, another table for customers, and another for stores. Each table comprises columns and rows (attributes and tuples in relational theory). Each column represents some attribute of the object represented by the table. For example, an **Employee** table would typically have columns for first name, last name, employee ID, department, pay grade, and job title. Each row represents an instance of the object represented by the table. For example, one row in the **Employee** table represents the employee who has employee ID 12345.

When organizing data into tables, you can usually find many different ways to define tables. Relational database theory defines a process, normalization, which ensures that the set of tables you define will organize your data effectively.

Task Define the following:

1. DBMS
2. Relational Database

Notes

1.4.3 Client/Server

In a client/server system, the server is a relatively large computer in a central location that manages a resource used by many people. When individuals need to use the resource, they connect over the network from their computers, or clients, to the server. Examples of servers are:

- ***Print servers:*** Manage the printers used by a team or unit.
- ***File servers:*** Store large files used by a team or unit by using large disk drives.
- ***E-mail servers:*** Run a company's e-mail system.

In a client/server database architecture, the database files and DBMS software reside on a server. A communications component is provided so applications can run on separate clients and communicate to the database server over a network. The SQL Server communication component also allows communication between an application running on the server and SQL Server.

Server applications are usually capable of working with several clients at the same time. SQL Server can work with thousands of client applications simultaneously. The server has features to prevent the logical problems that occur if a user tries to read or modify data currently being used by others.

Notes While SQL Server is designed to work as a server in a client/server network, it is also capable of working as a stand-alone database directly on the client.

The scalability and ease-of-use features of SQL Server allow it to work efficiently on a client without consuming too many resources.

1.4.4 Structured Query Language (SQL)

To work with data in a database, you must use a set of commands and statements (language) defined by the DBMS software. There are several different languages that can be used with relational databases; the most common is SQL. Standards for SQL have been defined by both the American National Standards Institute (ANSI) and the International Standards Organization (ISO). Most modern DBMS products support the Entry Level of SQL-92, the latest SQL standard (published in 1992).

Self Assessment

State true or false:

6. In a client/server database architecture, the database files and DBMS software reside on a client.
7. There are different ways to organize data in a database but relational databases are one of the most effective.
8. A database typically has two components.
9. Standards for SQL has been defined by the American National Standards Institute (ANSI) only.
10. SQL Server is capable of working as a stand-alone database directly on the client only.

Notes

1.5 SQL Server Databases

A database in SQL Server is made up of a collection of tables. These tables contain data and other objects, such as views, indexes, stored procedures, user-defined functions, and triggers that are defined to support activities performed with the data. The data stored in a database is typically related to a particular subject or process, such as inventory information for a manufacturing warehouse. You will learn about SQL server databases in detail in Unit 5.

1.6 Database Storage

SQL Server stores data and the log in disk files. In a basic installation, and as a default, data and log files are created in the default location specified in the server configuration. However, to maximize performance and manageability, you can apply a few basic principles:

- Spread data over as many disks, channels, and controllers as possible.

  In general, the more disks (spindles) you have (regardless of their individual size) and the faster your access to them (controllers and channels), the faster the storage engine can read and write data. The larger your system usage becomes, the more important it is to separate the data files from log files by storing them on different sets of physical drives. Also, because the use of tempdb has changed, you should now store tempdb on a large set of disks, for example, with the data files or on a set of disks.

- Use filegroups to make your enterprise database more manageable.

  Every database begins with one default filegroup. Because SQL Server 2000 can work effectively without additional filegroups, many systems will not need to add user-defined filegroups. However, as a system grows, the use of additional filegroups can provide more manageability, when implemented and maintained by a qualified DBA.

In SQL Server 2000, if you set a particular filegroup within a database to read-only, the data on that filegroup cannot be altered, but catalog information such as permissions can still be managed.

Notes In SQL Server 2000, the number of asynchronous I/Os is now managed dynamically inside the database engine, and is not influenced by the number of files or filegroups used, as was the case in SQL Server 7.0.

When implementing or optimizing a database design, the Database Administrator (Database System Engineer) needs to consider the configuration of the database storage components, particularly the layout of physical and logical disks and the arrangement of the database files across disks.

1.7 Security: Windows and SQL Server Authentication

Microsoft SQL Server can operate in one of two security (authentication) modes:

1. Windows Authentication Mode (Windows Authentication): Windows Authentication mode allows a user to connect through a Microsoft Windows NT® 4.0 or Windows® 2000 user account.
2. Mixed Mode (Windows Authentication and SQL Server Authentication): Mixed Mode allows users to connect to an instance of SQL Server using either Windows Authentication or SQL Server Authentication. Users who connect through a Windows NT 4.0 or Windows 2000 user account can make use of trusted connections in either Windows Authentication Mode or Mixed Mode.

Notes

SQL Server Authentication is provided for backward compatibility. For example, if you create a single Windows 2000 group and add all necessary users to that group you will need to grant the Windows 2000 group login rights to SQL Server and access to any necessary databases.

Notes You will study in detail about windows and SQL server authentication later in unit 6.

1.8 Summary

- SQL is a tool for organizing, managing, and retrieving data stored by a computer database. The acronym SQL is an abbreviation for Structured Query Language. SQL is thus a comprehensive language for controlling and interacting with a database management system.
- Microsoft makes SQL Server available in multiple editions, with different feature sets and targeting different users.
- Microsoft SQL Server is a Structured Query Language (SQL) based, client/server relational database.
- A database typically has two components: the files holding the physical database and the database management system (DBMS) software that applications use to access data.
- There are several different languages that can be used with relational databases; the most common is SQL. Standards for SQL have been defined by both the American National Standards Institute (ANSI) and the International Standards Organization (ISO).
- Microsoft SQL Server can operate in one of two security (authentication) modes: windows authentication mode and Mixed mode.

1.9 Keywords

Database: A collection of information, tables, and other objects organized and presented to serve a specific purpose, such as searching, sorting, and recombining data. Databases are stored in files.

Microsoft SQL Server: Microsoft SQL Server is a Structured Query Language (SQL) based, client/server relational database.

Relational Database: Relational database systems are an application of mathematical set theory to the problem of effectively organizing data. In a relational database, data is collected into tables.

SQL: It is a tool for organizing, managing, and retrieving data stored by a computer database.

1.10 Review Questions

1. Define SQL Server. Also explain its functions.
2. Discuss the various mainstream editions of SQL server.
3. Briefly describe the specialized editions of SQL server.
4. Describe SQL server architecture briefly.
5. Discuss the two security modes in SQL server.

Notes

6. Explain the concept of SQL. Also discuss its uses.
7. Explain: Client Server database architecture.

Answers: Self Assessment

1.	SQL Server	2.	SQL Server web edition
3.	Microsoft SQL Azure Database	4.	SQL Server 2008 R2 Datacenter
5.	SQL Server Enterprise Edition	6.	False
7.	True	8.	True
9.	False	10.	False

1.11 Further Readings

Books

C.J. Date, *Introduction to Database Systems*, Pearson Education.

Elmasri Navrate, *Fundamentals of Database Systems*, Pearson Education.

Peter Rob & Carlos Coronel, *Database Systems Design, Implementation and Management*, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, *Database Management Systems*, Tata McGraw Hill, 3rd Edition.

Silberschatz, Korth, *Database System Concepts*, McGraw Hill, 5th Edition.

Online links

www.sql-tutorial.net/

www.w3schools.com/sql/

www.sqlservertutorials.com/

Notes

Unit 2: Installing SQL Server

CONTENTS

Objectives

Introduction

2.1 Hardware and Software Requirements for SQL Server 2008 Installation

2.2 Procedure of Installing Microsoft SQL Server

2.3 Installing the Clients Tools of Microsoft SQL Server

2.4 Using SQL Server

2.4.1 Opening MS SQL Server

2.4.2 Launching SQL Server

2.5 Security Considerations for a SQL Server Installation

2.5.1 Before Installing SQL Server

2.5.2 After Installing SQL Server

2.6 Summary

2.7 Keywords

2.8 Review Questions

2.9 Further Readings

Objectives

After studying this unit, you will be able to:

- Know the minimum hardware and software requirements to install and run SQL Server 2008
- Understand the procedure of installing SQL server
- Describes security considerations for a SQL Server installation

Introduction

There are various ways of installing sql server. The primary type consists of installing Microsoft SQL Server on only one computer you will use to do everything. That machine may not be connected to another or you will not be concerned with using things (files or databases) in other computers. With this type of setup, you perform the installation on that computer and simply start using Microsoft SQL Server.

The SQL Server Installation Wizard is Windows Installer-based. It provides a single feature tree to install all SQL Server components:

- SQL Server Database Engine
- Analysis Services
- Reporting Services
- Integration Services
- Replication

Notes

- Management tools
- Documentation

Notes SQL Server is available in 32-bit and 64-bit editions. The 64-bit and 32-bit editions of SQL Server are installed either through the Installation Wizard, or at a command prompt.

2.1 Hardware and Software Requirements for SQL Server 2008 Installation

The following sections list the minimum hardware and software requirements to install and run SQL Server 2008.

The following requirements apply to all SQL Server 2008 installations:

1. ***Framework:*** SQL Server Setup installs the following software components required by the product:
   - NET Framework 3.5
   - SQL Server Native Client
   - SQL Server Setup support files
2. ***Software:*** SQL Server Setup requires Microsoft Windows Installer 4.5 or a later version, and Microsoft Data Access Components (MDAC) 2.8 SP1 or a later version.
3. ***Internet Software:*** Microsoft Internet Explorer 6 SP1 or a later version is required for all installations of SQL Server 2008. Internet Explorer 6 SP1 or a later version is required for Microsoft Management Console (MMC), SQL Server Management Studio, Business Intelligence Development Studio, the Report Designer component of Reporting Services, and HTML Help.
4. ***Display:*** SQL Server 2008 graphical tools require VGA or higher resolution: at least 1,024×768 pixel resolution.

For Installing SQL Server 2008 Enterprise Edition:

1. ***Processor:*** Processor type: Pentium III-compatible processor or faster

   Processor speed: Minimum: 1.0 GHz, Recommended: 2.0 GHz or faster
2. ***Operating System:*** Windows XP Professional SP2
3. ***Memory:***

   RAM:

   Minimum: 512 MB

   Recommended: 2.048 GB or more

   Maximum: Operating system maximum
4. ***Features and Disk Space Requirement:***

Database Engine and data files, Replication, and Full-Text Search	: 280 MB
Analysis Services and data files	: 90 MB
Reporting Services and Report Manager	: 120 MB

Notes

Integration Services	:	120 MB
Client Components	:	850 MB

Notes SQL Server 2008 is not supported on Windows Server 2008 and Windows Server 2008 R2 Server Core installations.

Task List the hardware and software requirements of various editions of SQL server 2008.

2.2 Procedure of Installing Microsoft SQL Server

Follow the below given steps for installing SQL Server.

1. Start the computer
2. Log in using an account with administrative rights. If you followed the previous lesson, log in using the phkatts account. If the computer you are using is connected to a network, make sure you specify the domain. If you are logged in locally, make sure you use the name of the computer.

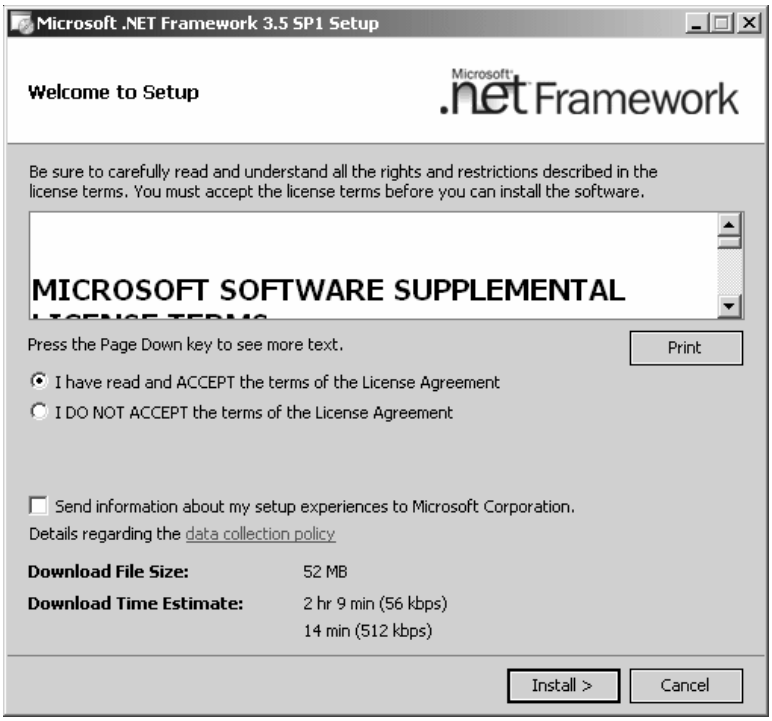
   If you are installing in Microsoft Windows XP-7, make sure you log in with a password, otherwise the installation will fail.

   For our installation, we will log in as Administrator (in either Microsoft Windows 7 Professional/Ultimate or in a domain named functionx.local)
3. Insert the DVD in the drive. You may (should) receive a dialog box with a Run SETUP.EXE link:

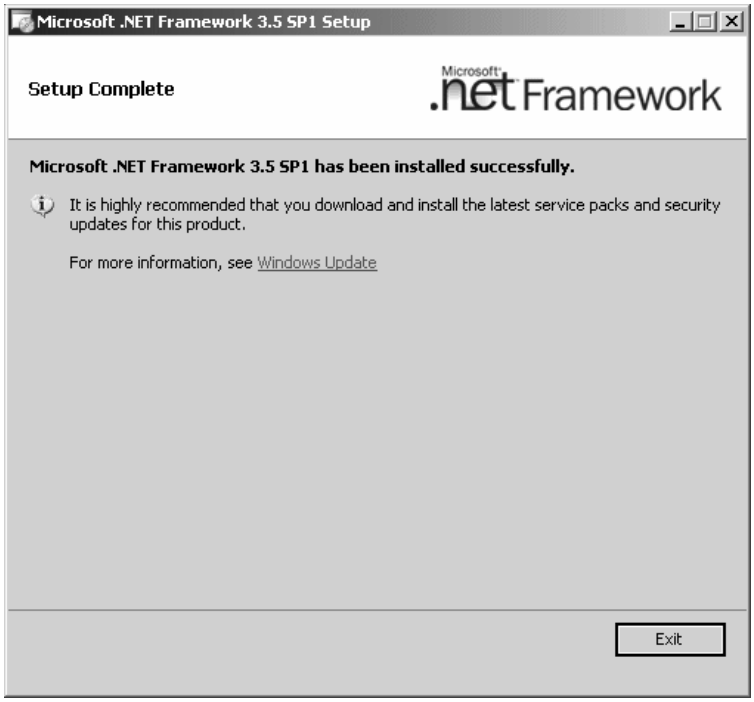
If so, click Run SETUP.EXE

Notes

4. The installation will start. You may get a message that the .NET Framework needs to be installed. If you get that message, click OK. You will be prompted to start the installation:

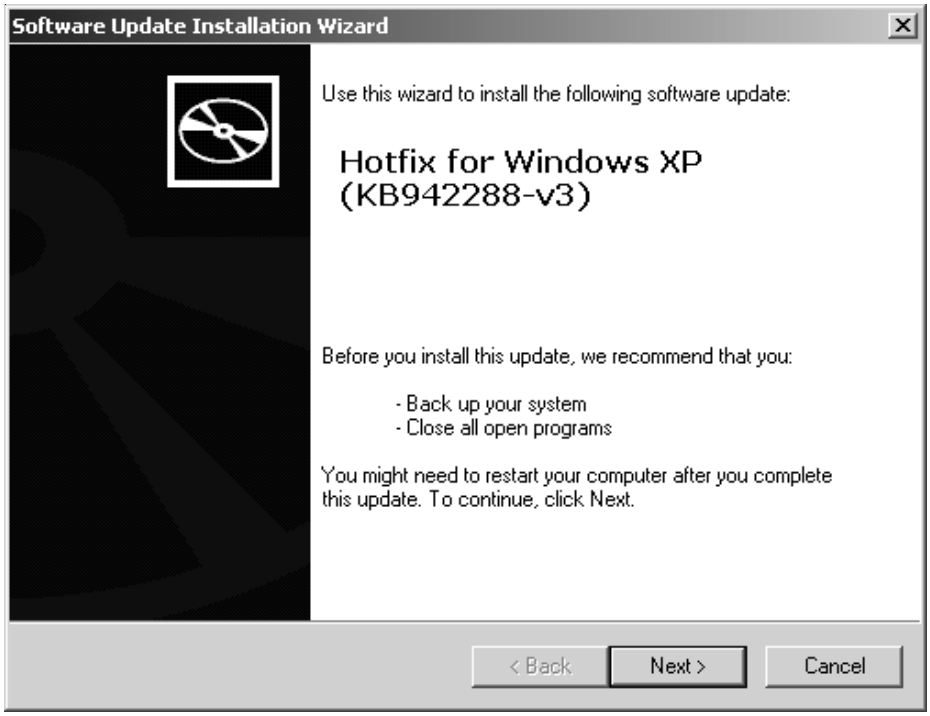


After clicking the ACCEPT button, you can click Install. The installation of the .NET Framework would start. When the installation of the .NET Framework is over, you would receive a message:

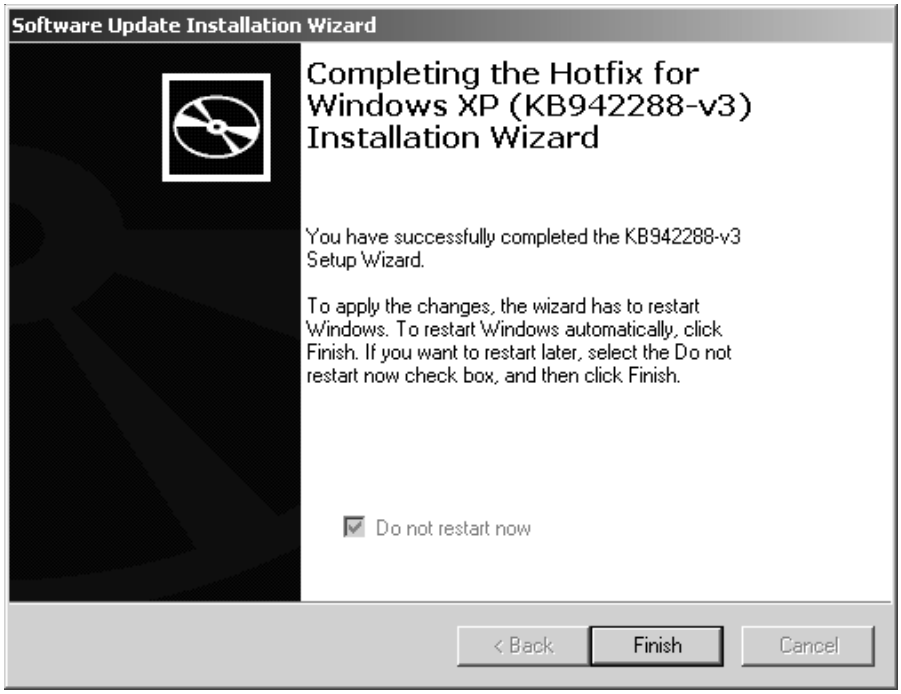


Notes

From there, you can click Exit. If you are installing in either Microsoft Windows XP or Microsoft Windows Server 2003, you may be asked to install a hotfix:



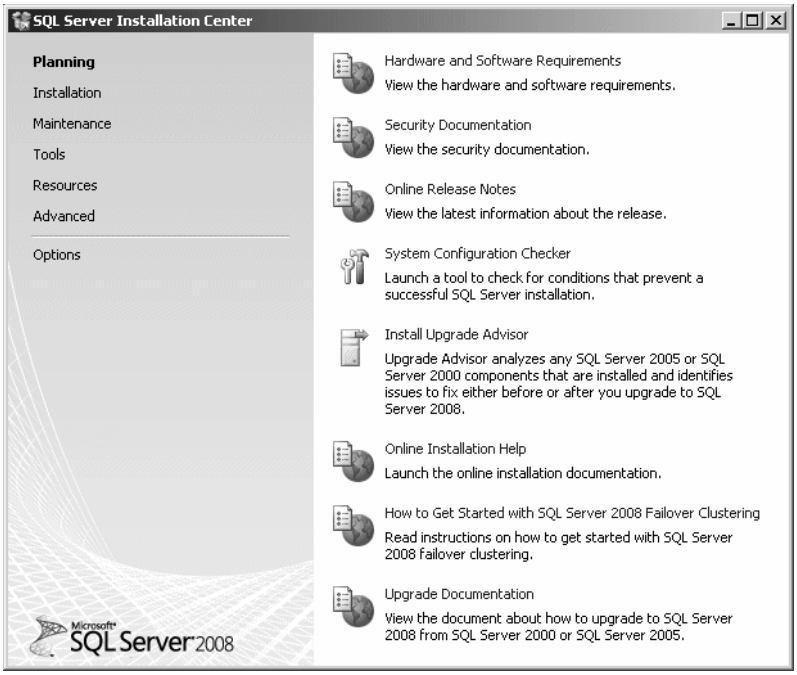
In this case, accept to install it by clicking Next and following the wizard. When the installation is over, you may be asked to restart:



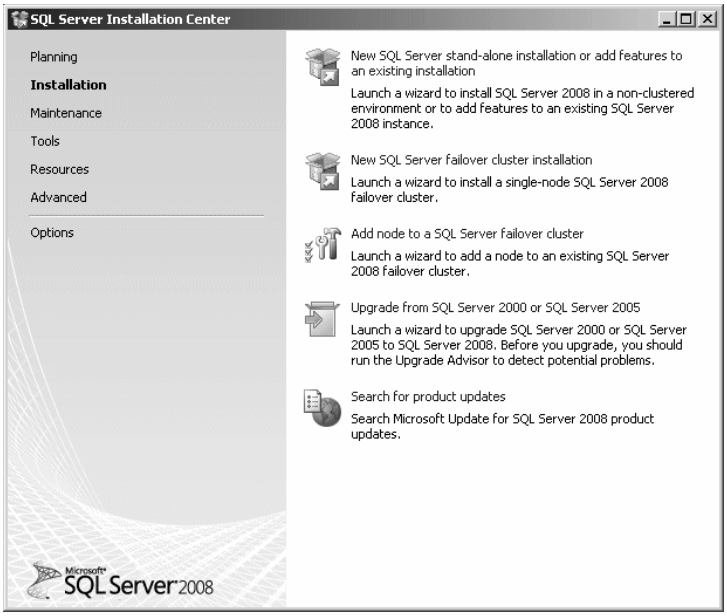
Which you should do.

Notes

5. To start the installation, put the Microsoft SQL Server 2008 DVD in the drive. A window would come up:



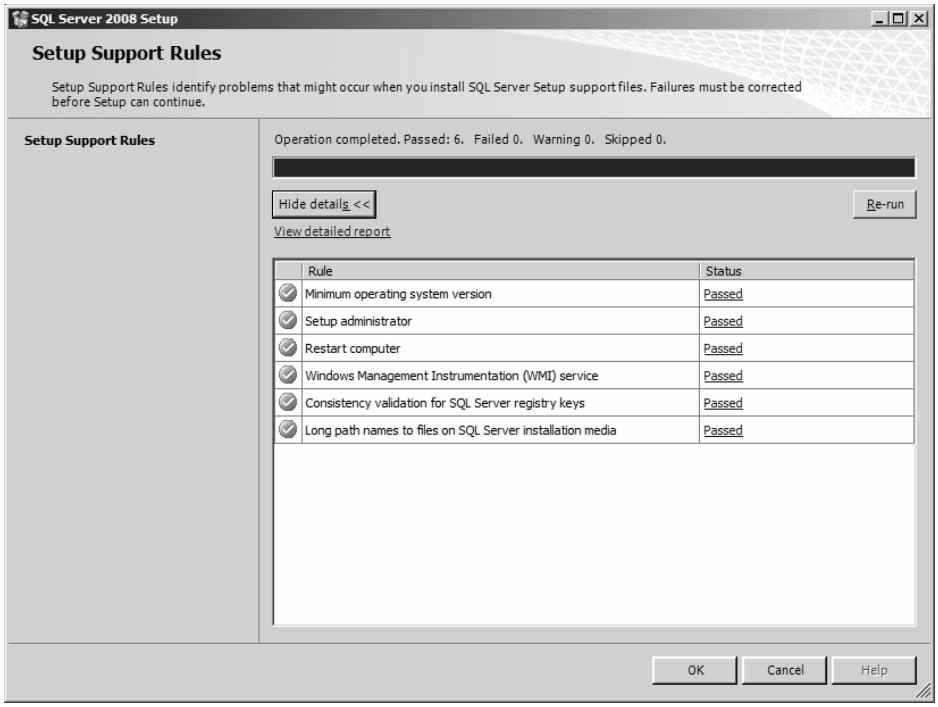
6. In the left section, click Installation



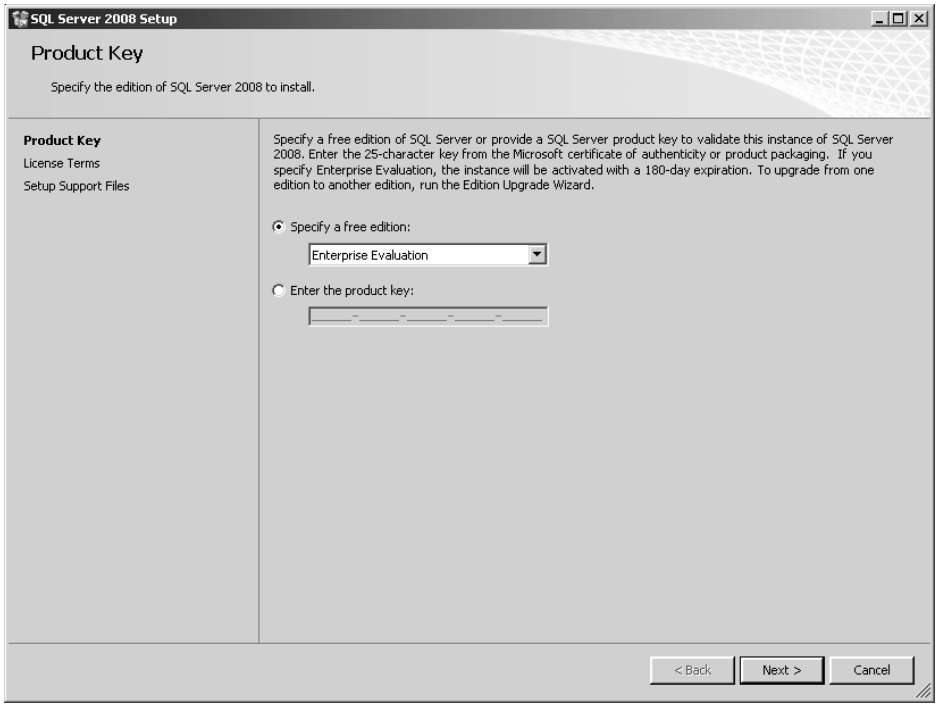
7. If you are only interested in studying database design and development in Microsoft SQL Server, on the right side, click New SQL Server Stand-Alone... The installation would start:

Notes

8. After the message box, click OK

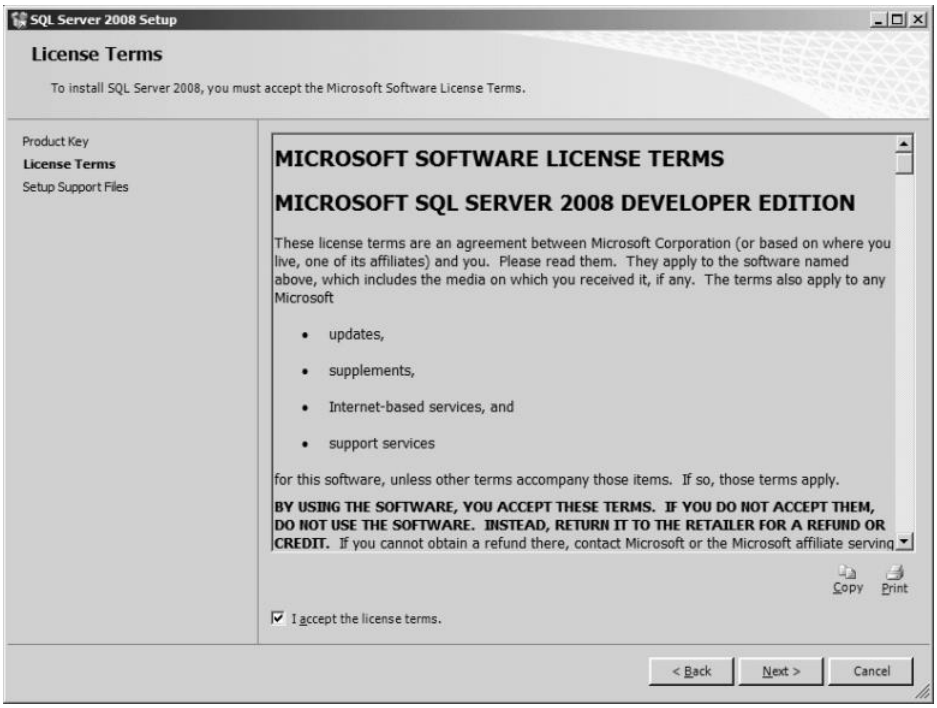


9. Another message box would come up. When it finishes, the next page of the wizard would come up. If you are using the evaluation version, read the message and click the first radio button.



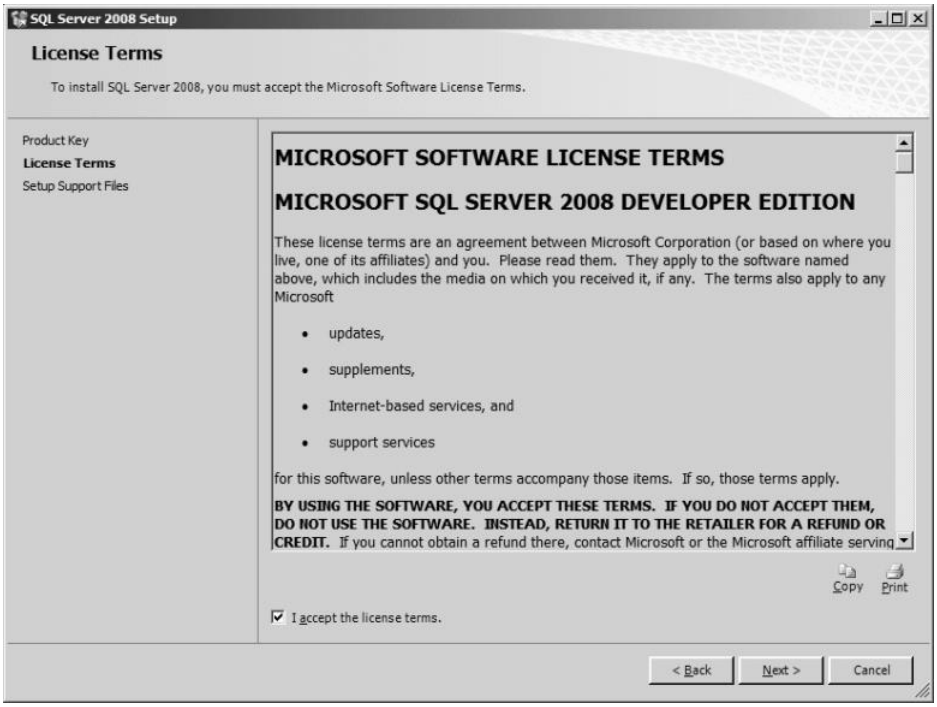
If you are installing with a product key, click the second radio button and enter the key

Notes



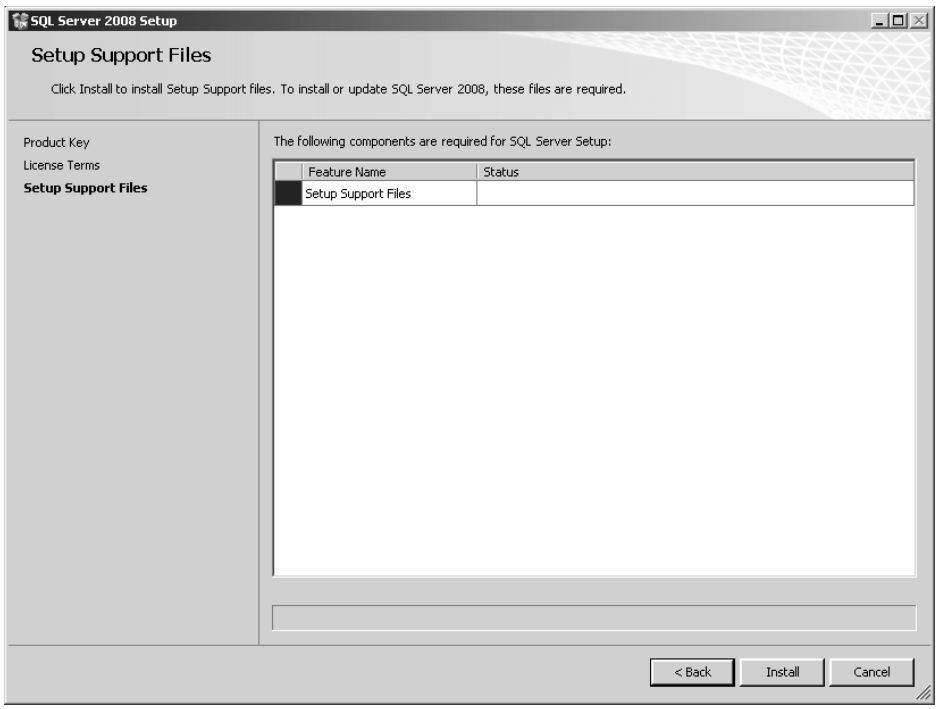
10. Click Next

11. The License Terms page is presented to you. Read it. If you don't agree with the license, click Cancel. Otherwise, click the I Accept check box:

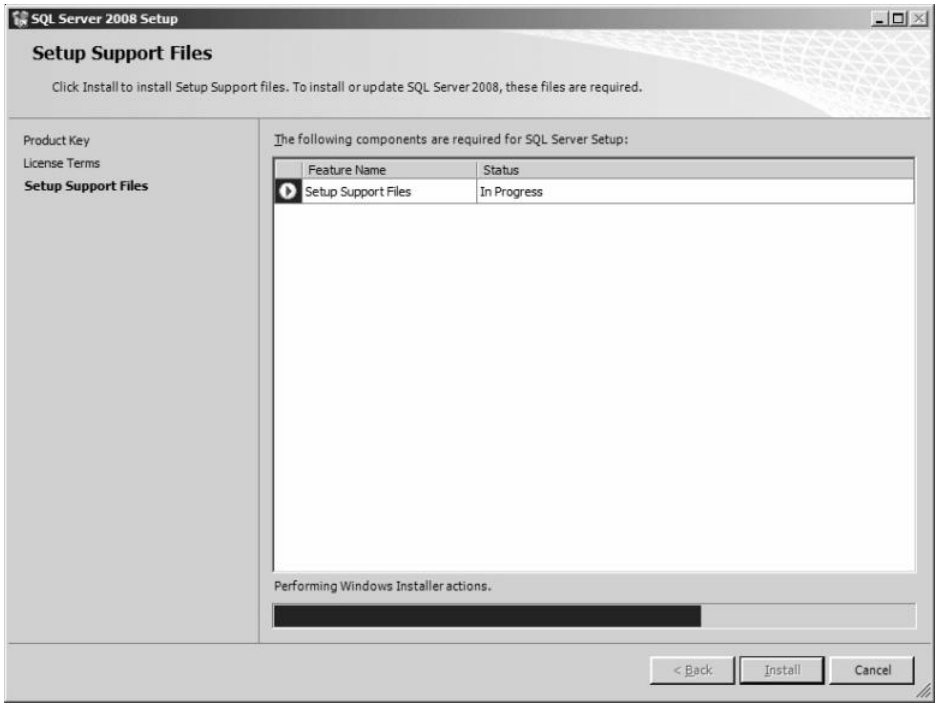


Notes

12. Click Next

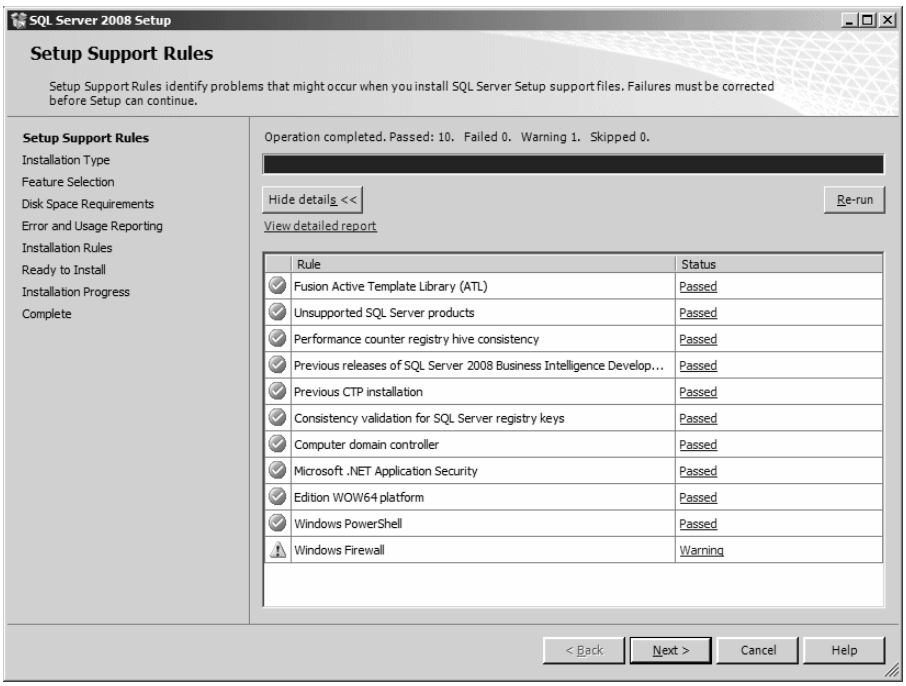


13. Read the messages and click Install:



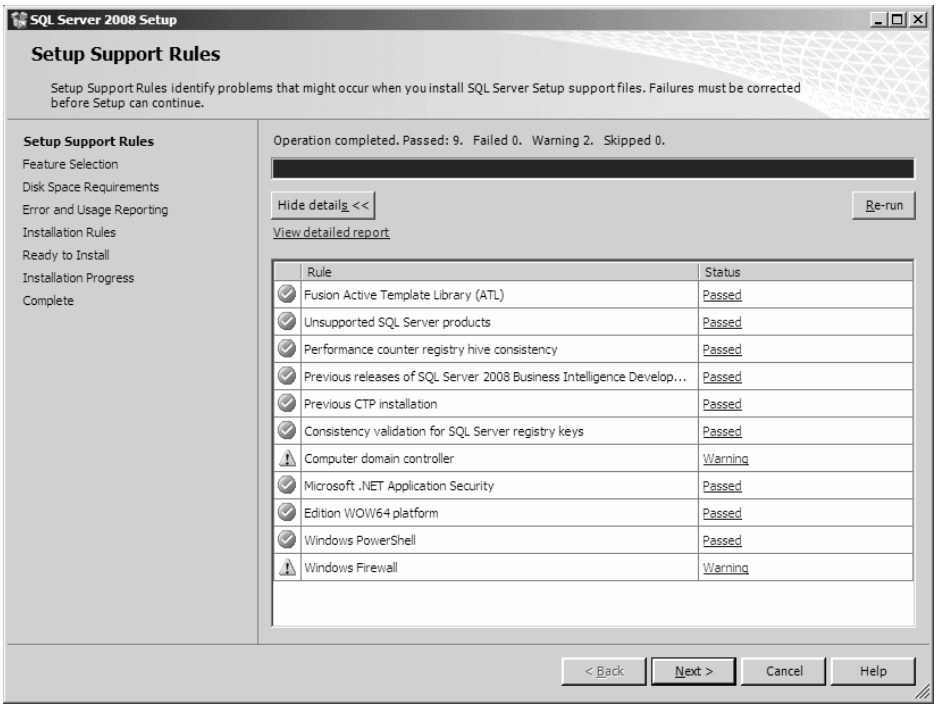
Notes

14. You may receive a warning message about the firewall:



For now, ignore it (there is no true justification in trying to disable the firewall, the installation will proceed fine).

If you are installing on the only computer you use as your server, you may receive a warning that it is not recommended to install Microsoft SQL Server on a domain controller:



Ignore the warning for now.

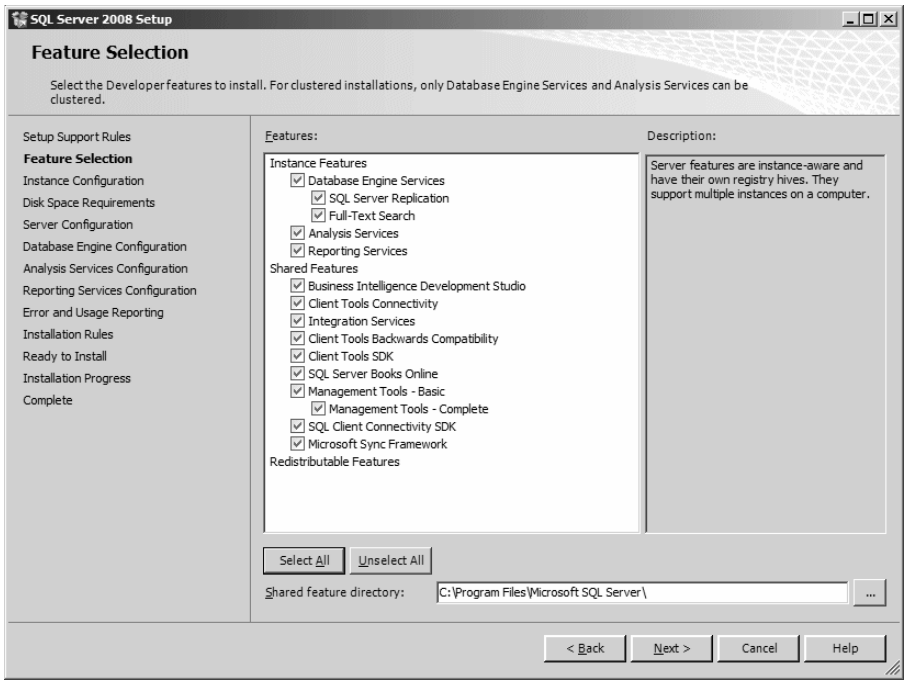
Click Next

Notes

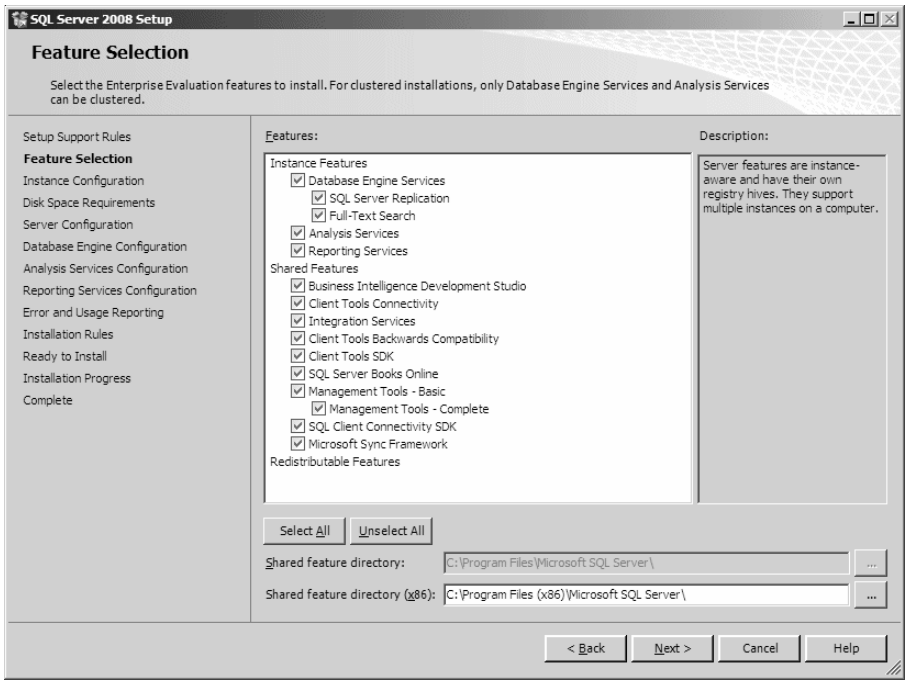
15. If you had previously installed Microsoft SQL Server Express, you may receive a message box asking you to specify an instance name. If so, specify it as MSSQLSERVER.

    The next page allows you to specify what you want to install and what would be left out. For our example, we select all, then removed Reporting Services.

    The eventual location of Microsoft SQL Server is specified in the bottom text box:



If you are installing the Developer Edition and if you to change the directory of Microsoft SQL Server, you can type it in the Shared Feature Directory text box, or you can click the browse button to select it.

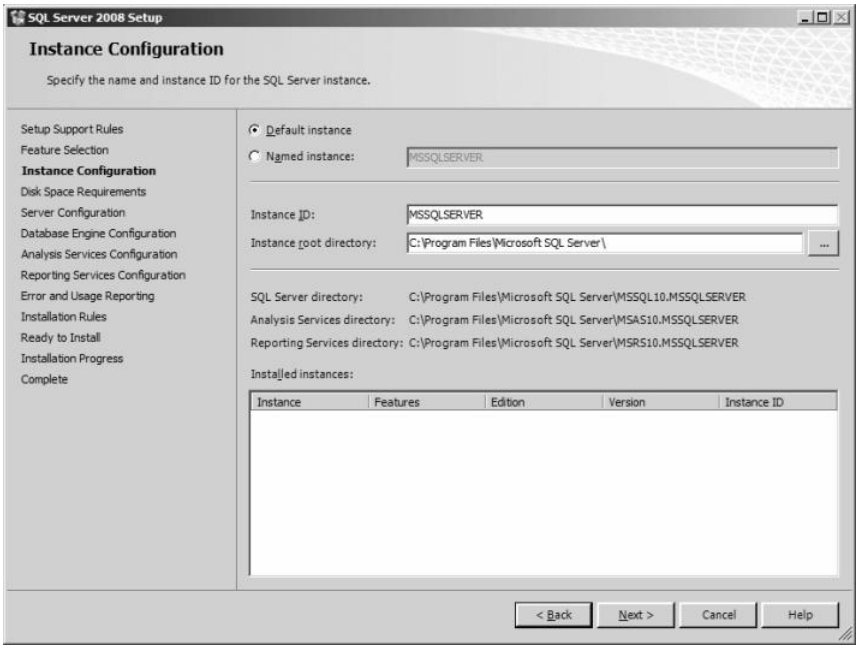


Notes

If you are installing the Enterprise Edition, if you to change the directory of Microsoft SQL Server, click the browser button on the right side of the Shared Feature Directory text box and locate the directory you want.

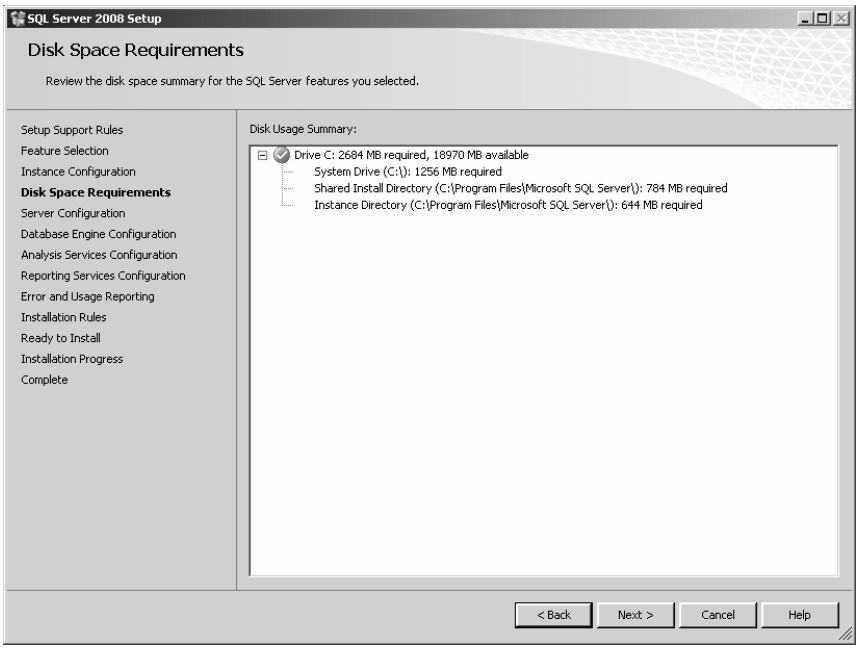
After making your selections, click Next

16. The next page allows you to name the instance of Microsoft SQL Server (and to specify where the server would be installed). In most cases, you should accept the suggested name of the instance as MSSQLSERVER:



After making your selections, click Next

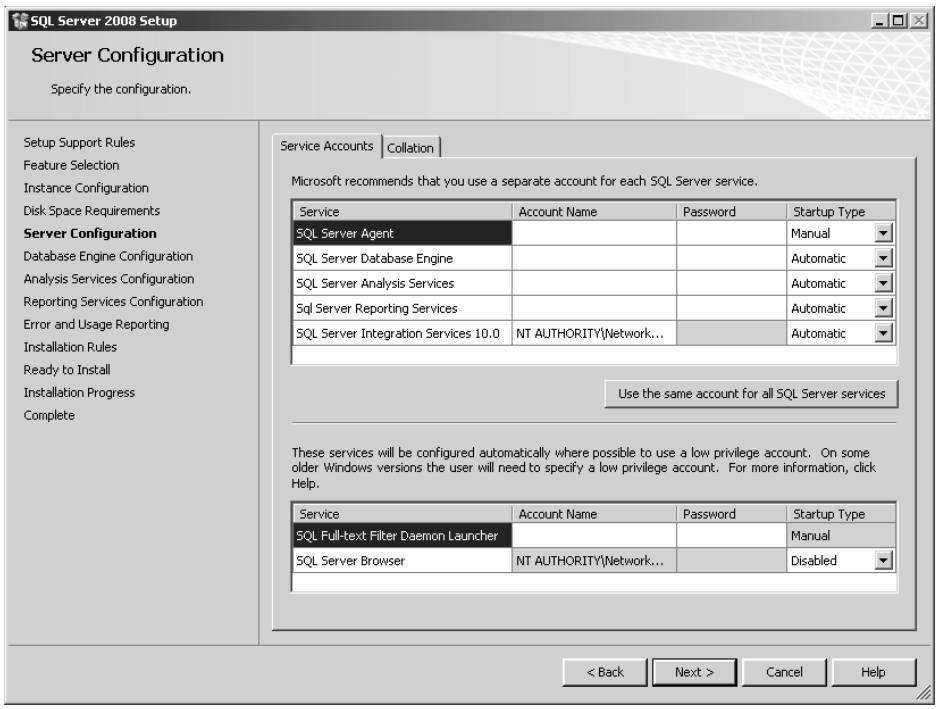
17. The next page mostly allows you to check and confirm the disk space:



After checking it, click Next

Notes

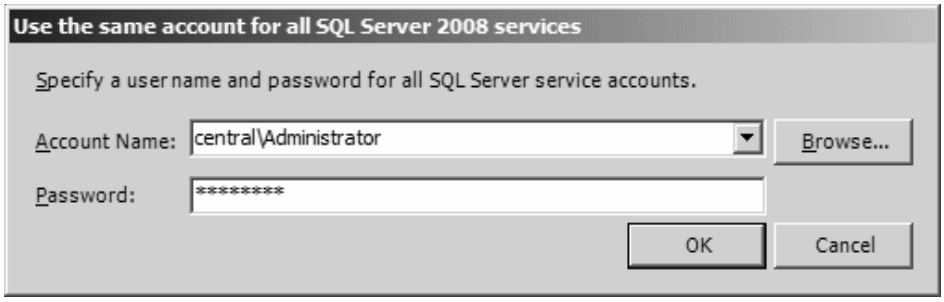
18. The next page allows you to specify the account that is performing the installation:



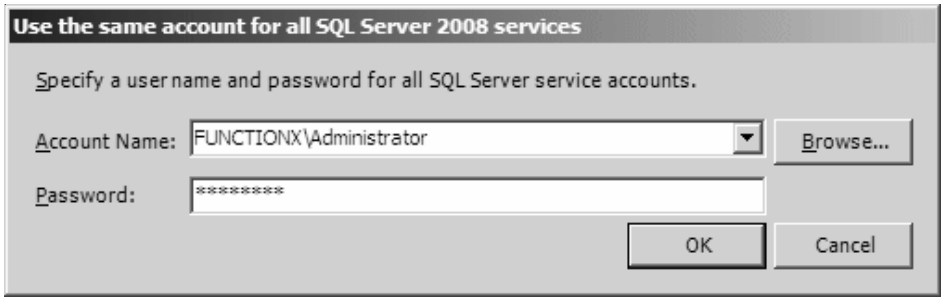
Click Use the Same Account For All Server Services

19. A dialog box would come up.

If you are installing in Microsoft Windows 7 and you want to use a local account, in the Account Name text box, enter the name of the computer, followed by a back slash, followed by the user name, and press Tab. In the Password text box, enter the password:

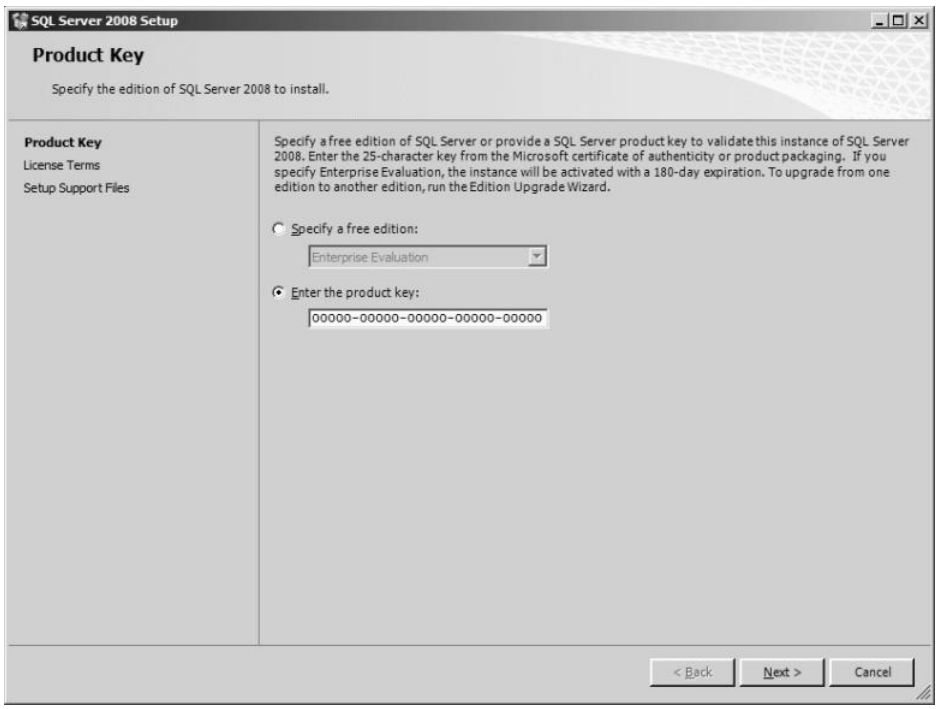


If you are installing in Microsoft Windows Server 2008, in the Account Name text box, enter the name of the domain, followed by a back slash, followed by the user name, and press Tab. In the Password text box, enter the password:

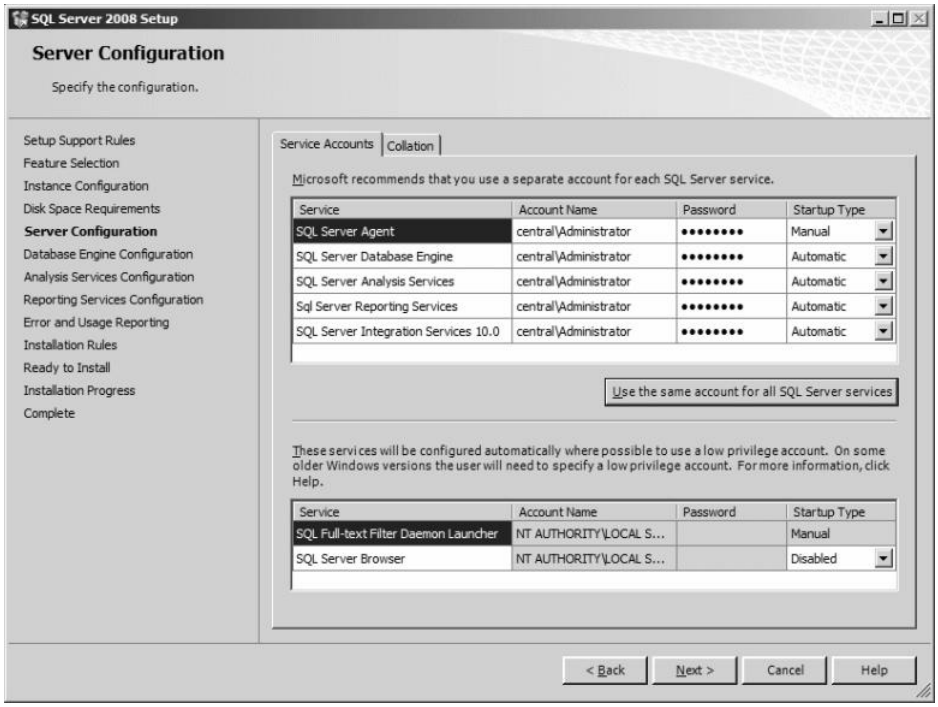


Notes

If you are installing with a product key, click the second radio button and enter the key

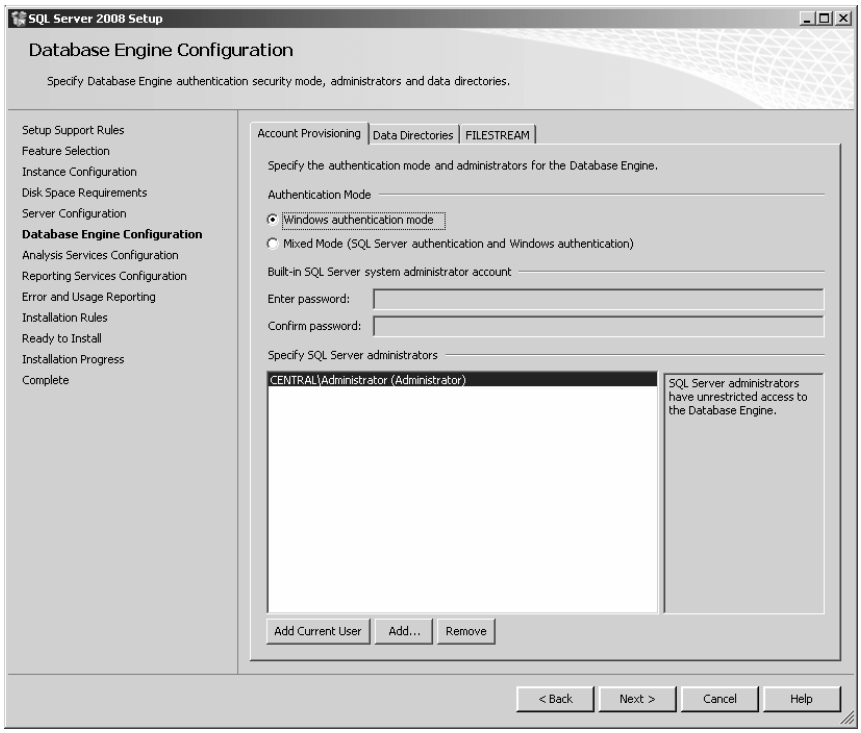


20. Click Next



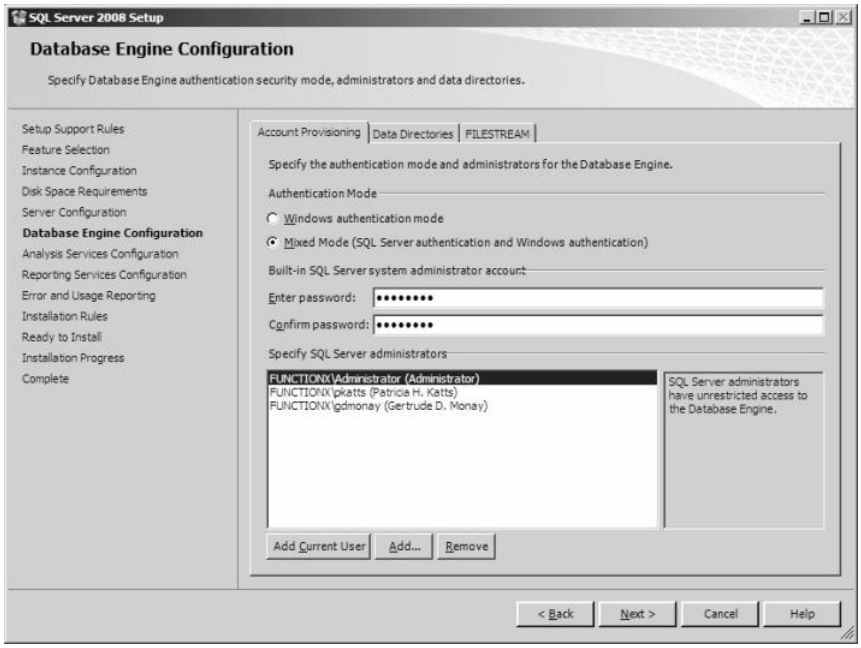
21. Click Next
22. After making your selections, click Next
23. In the next page, specify how authentication would be made. You should also specify the account used as the administrator. To do this, you can click Add Current User:

Notes



24. Microsoft SQL Server provides a default user named sa. To prepare that user for database authentication, click the Mixed Mode
25. Type a password in the Enter Password text box. For this exercise, you can use **P@assword1** and press Tab
26. Enter the same password in the Confirm Password text box
27. To add an additional account, click Add...

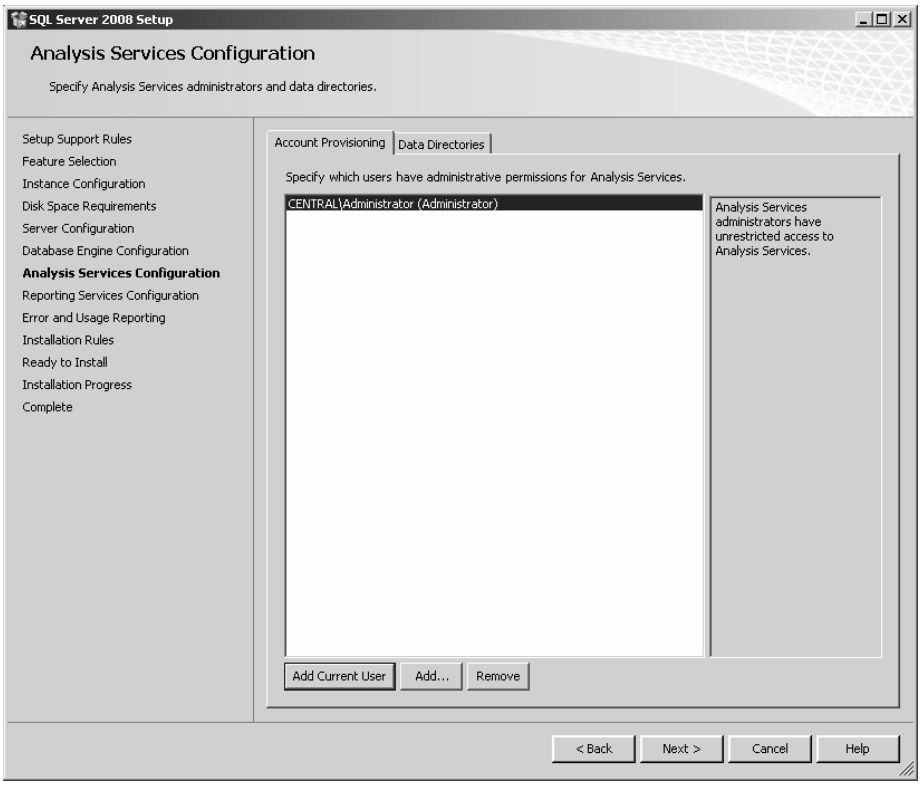
    A dialog box would display. In the bottom text box, type the (complete) user name of a user and click Check Names. In the same way, you can add as many users as possible



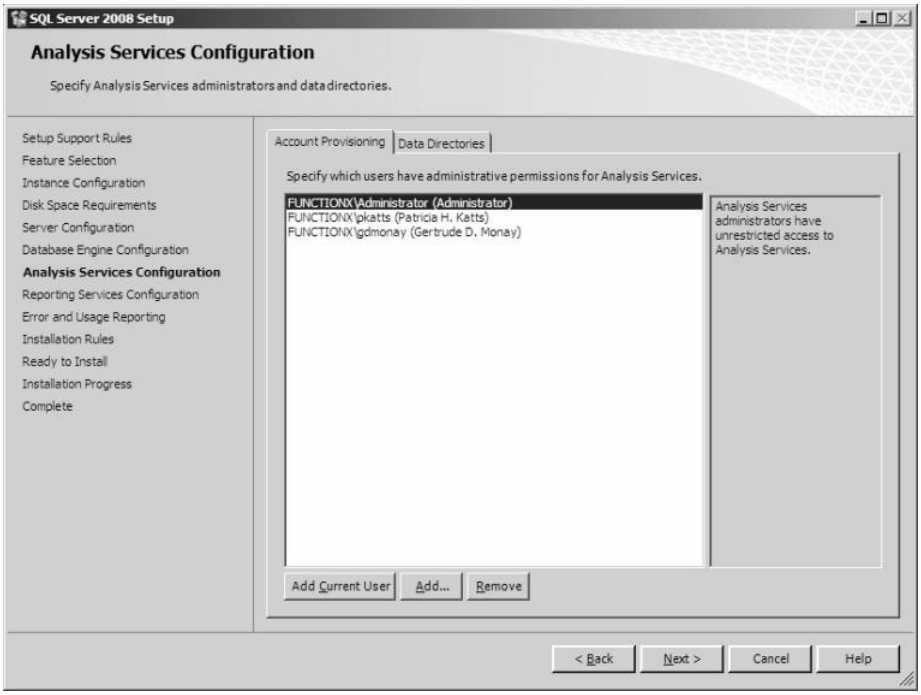
Notes

28. Click Next

29. The next step allows you to specify an account for the analysis services. Click Add Current User



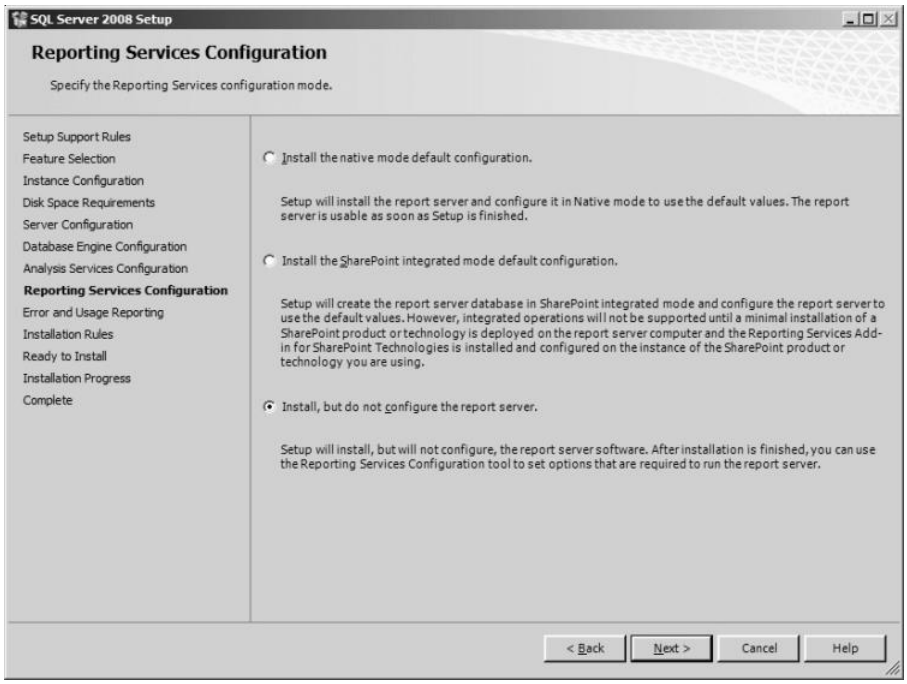
To add an additional account, click Add... In the bottom text box, type the (complete) user name of a user and click Check Names. In the same way, you can add as many users as possible



Notes

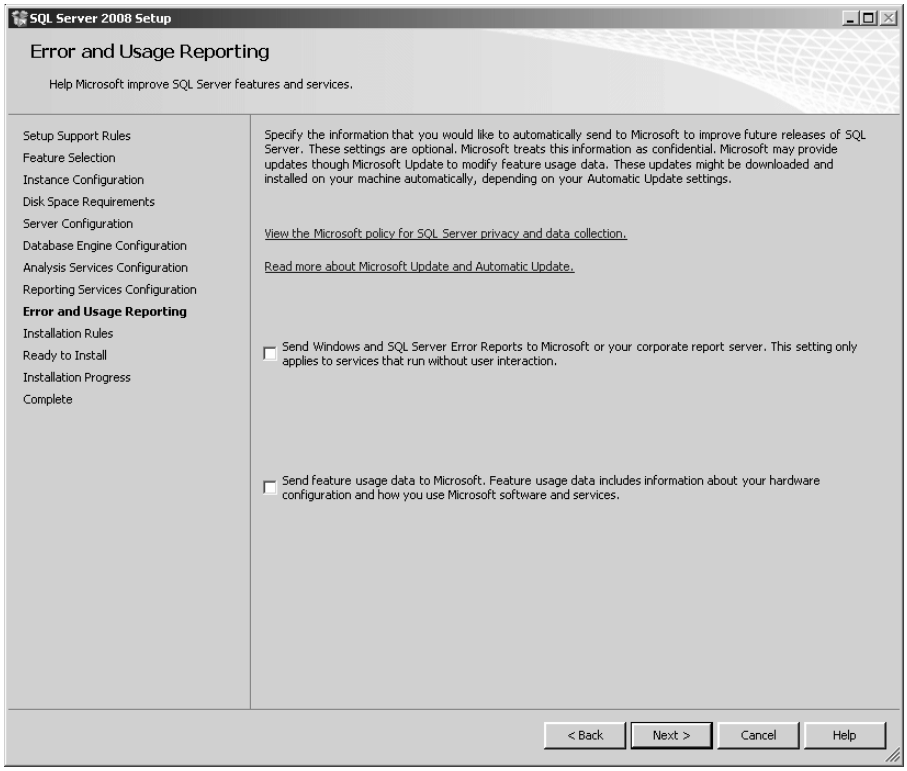
30. Click Next

31. In the Reporting Services Configuration page, click the last radio button to install but without configuring the reporting:



32. Click Next.

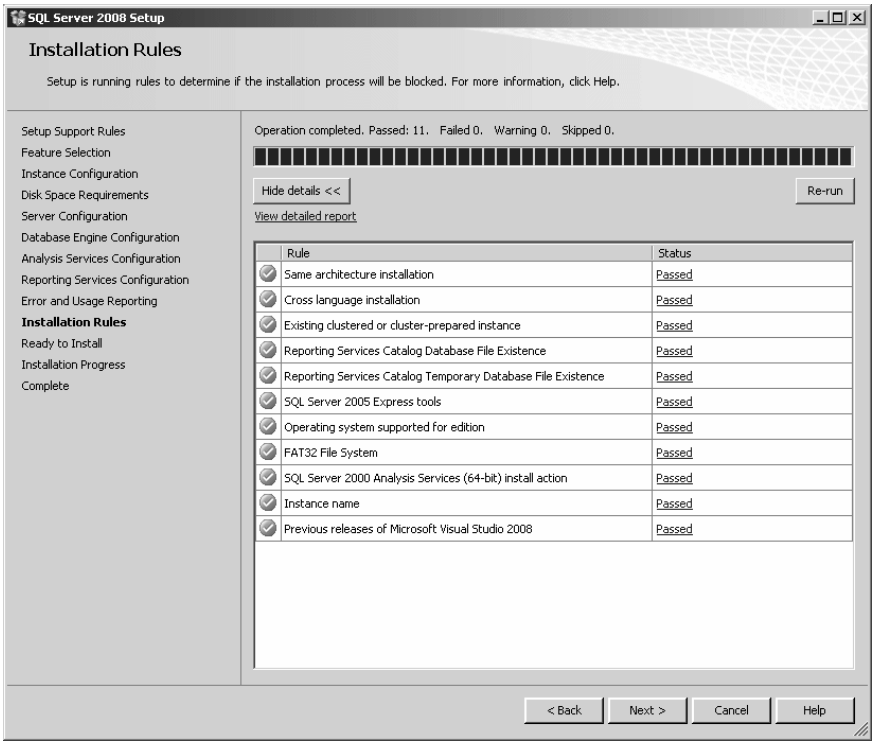
The next page allows you to specify whether you want and installation report to be sent to Microsoft:



Notes

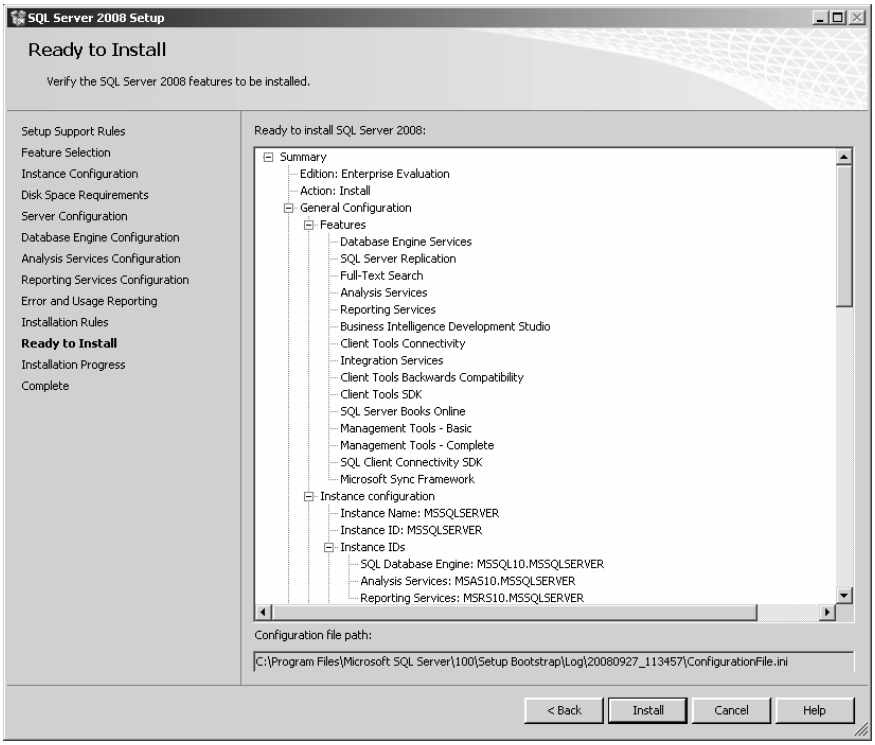
33. Read it and click Next

34. The next page gives you a summary of tests that were performed before the installation:



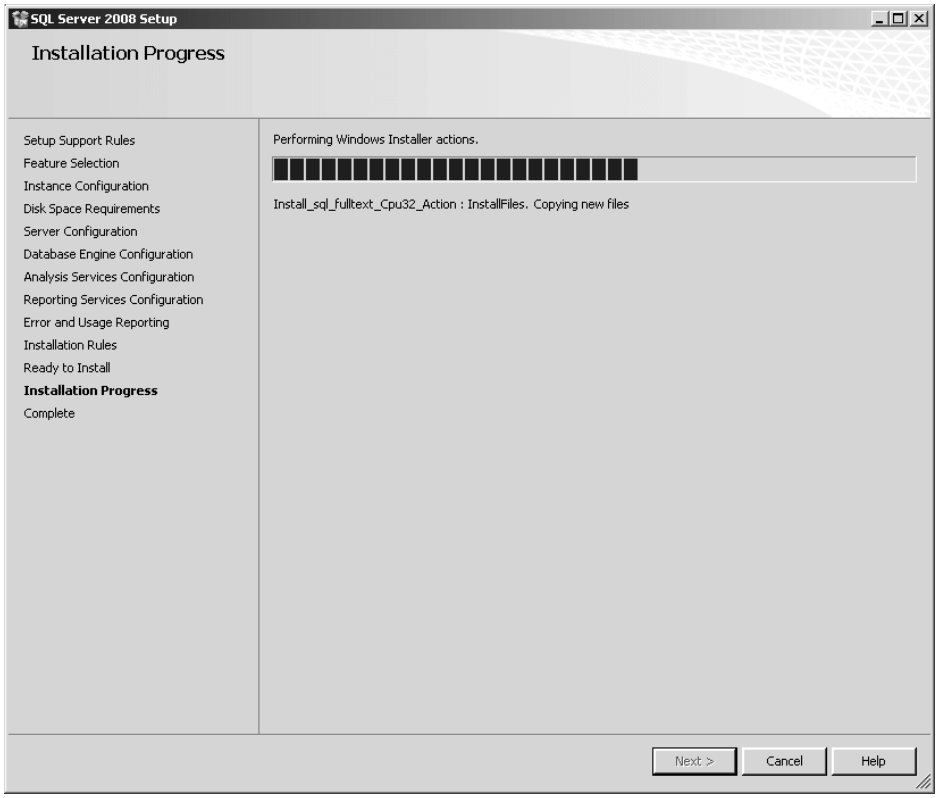
35. After reading, click Next.

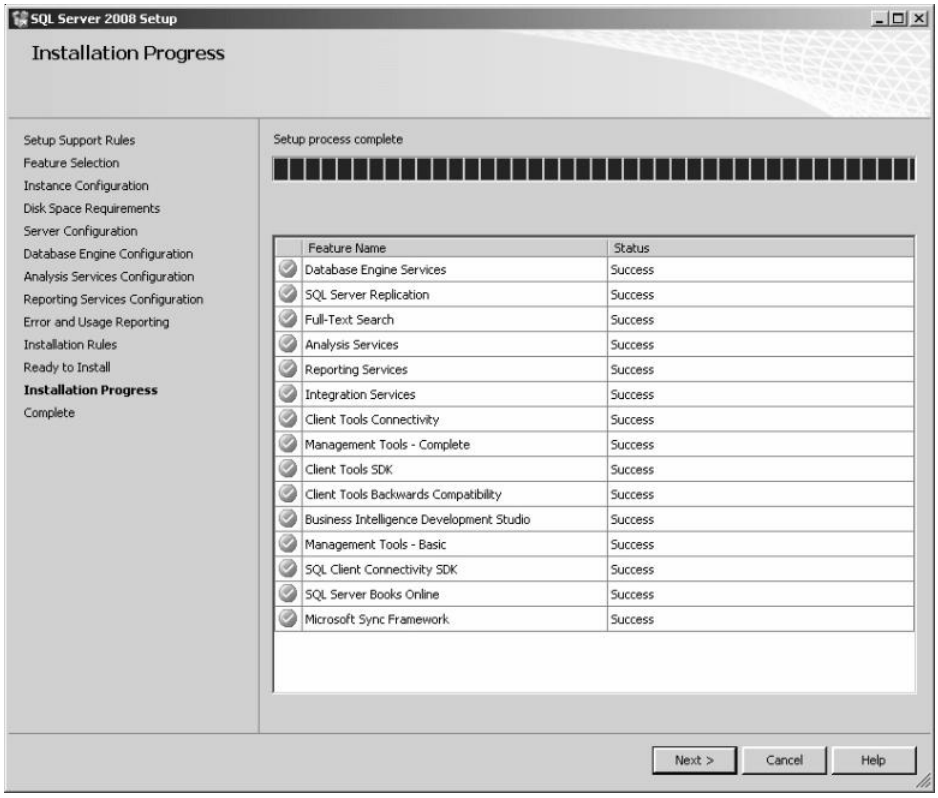
This would indicate that the installation is ready to proceed:



Notes

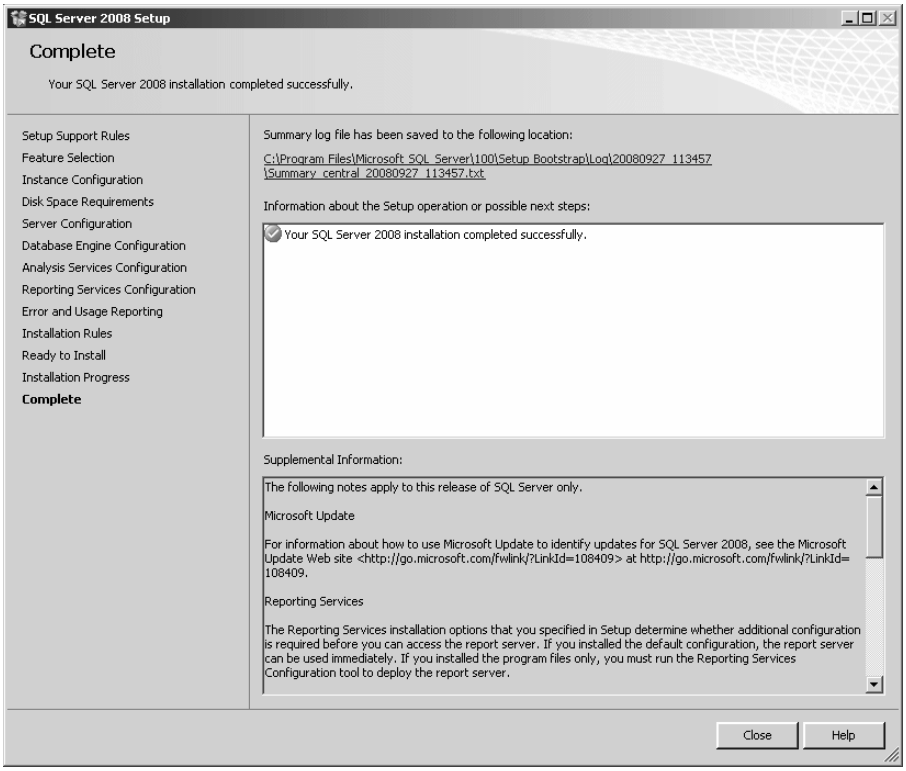
36. To start the installation, click Install. The installation would start and a progress bar would indicate the evolution. This can take a while





Notes

37. When this phase is over, a message box will let you know. After reading it, you can click Next. The last page of the wizard should announce that the installation was complete



38. After reading the message, click Close

Task List the various steps of installing SQL server on a stand-alone computer

Self Assessment

Fill in the blanks:

1. The SQL Server Installation Wizard is ..... Installer-based. It provides a single feature tree to install all SQL Server components:
2. There are ..... ways of installing sql server.
3. SQL Server Setup requires Microsoft Windows Installer ..... or a later version, and Microsoft Data Access Components (MDAC) 2.8 SP1 or a later version.
4. Microsoft Internet Explorer ..... or a later version is required for all installations of SQL Server 2008.
5. SQL Server 2008 graphical tools require ..... or higher resolution: at least 1,024 × 768 pixel resolution.

2.3 Installing the Clients Tools of Microsoft SQL Server

You may have Microsoft SQL Server installed in a computer protected (locked) somewhere in a different room or in a different building, or you may be working in a networked environment

Notes

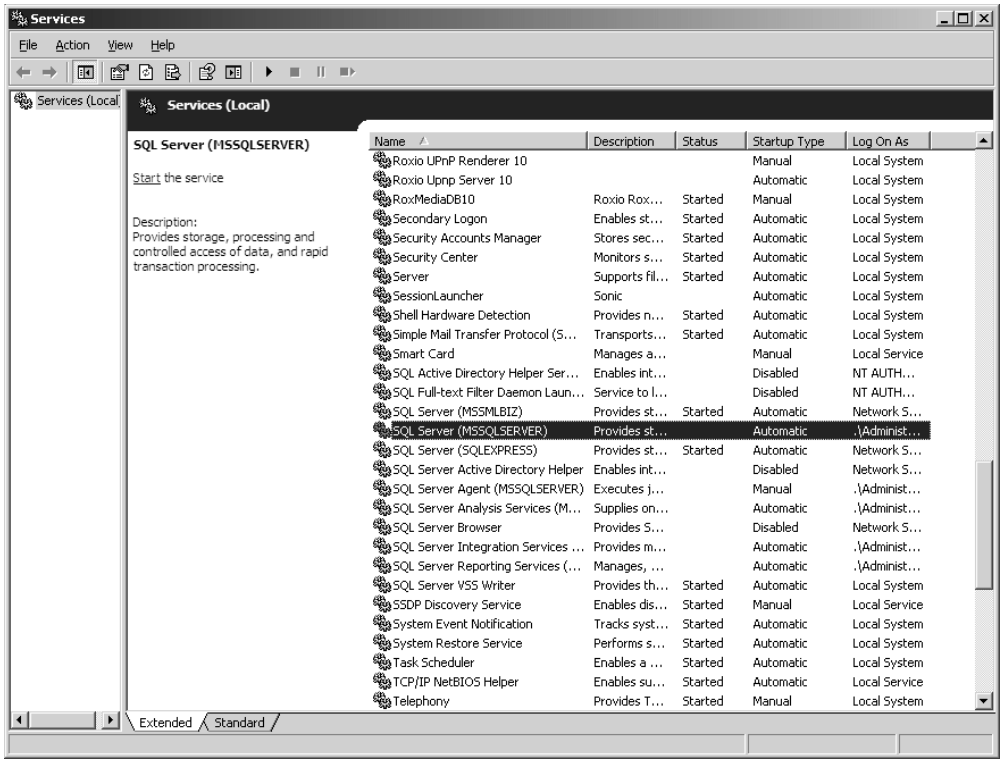
where computers interact with each other, or you may be asked to test something on someone else's computer that has Microsoft Server but you must not modify anything on that machine. To satisfy this and other requirements, you can install a suite of applications that would allow you to access Microsoft SQL Server installed in a computer other than the one you are using. Such tools are referred to as client tools.

To install the client tools of Microsoft SQL Server, you use the same DVD of the server installation but make different choices.

2.4 Using SQL Server

After installing Microsoft SQL Server, you can start using it. Because Microsoft SQL Server works as a service to the operating system, in order to use it, you must make sure its service has started. Sometimes you will have to start the service and sometimes you will have to stop it. To check it, you can open the Control Panel and the Administrative Tools. In the Administrative Tools window, open the Services.

In the Services window, check the status of the SQL Server (MSSQLSERVER), the SQL Server Agent (MSSQLSERVER), and the SQL Server Browser:

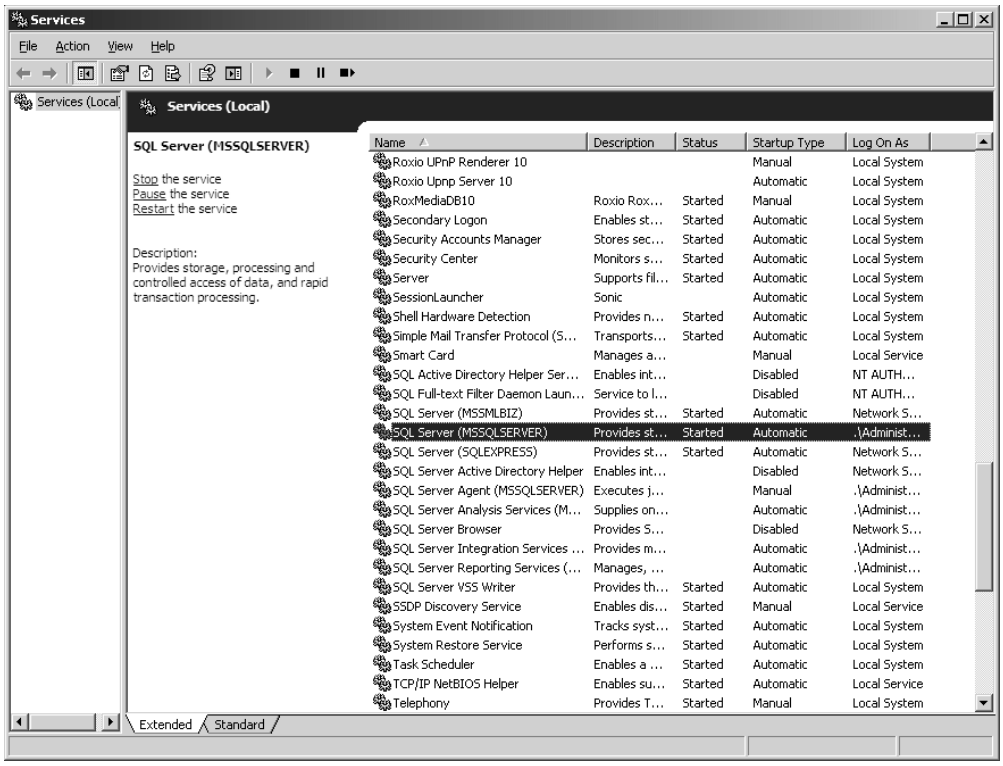


If the MSSQLSERVER service is stopped, you should start it. To do this, you can right-click it and click Start. If it fails to start, check the account with which you logged in:

- If you are using Microsoft Windows XP-7 and you logged in as Administrator but did not provide a password, you should open Control Panel, access User Accounts, open the Administrator account, and create a password for it
- If you are using a server (Microsoft Windows Server 2003 or Microsoft Windows Server 2008), make sure you logged in with an account that can start a service

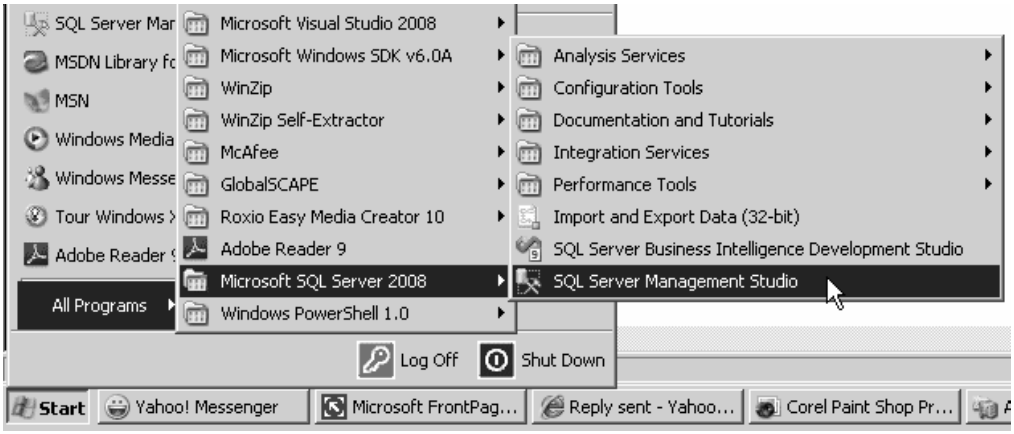
Once the service has started, it should be labeled Started:

Notes



2.4.1 Opening MS SQL Server

To launch Microsoft SQL Server, you can click Start → (All) Programs → Microsoft SQL Server 2008 → SQL Server Management Studio



When it starts, it would present a dialog box that expects you to log in.

2.4.2 Launching SQL Server

1. Start the computer
2. Log in with the account you used to install Microsoft SQL Server

Notes

3. To launch Microsoft SQL Server, click Start → (All) Programs → Microsoft SQL Server 2008 → SQL Server Management Studio. A splash screen will appear:

4. On the Connect to Server dialog box, click Cancel.

2.5 Security Considerations for a SQL Server Installation

Security is important for every product and every business. By following simple best practices, you can avoid many security vulnerabilities. This topic discusses some security best practices that you should consider both before you install SQL Server and after you install SQL Server. Security guidance for specific features is included in the reference topics for those features.

2.5.1 Before Installing SQL Server

Follow these best practices when you set up the server environment:

- Enhance physical security
- Use firewalls
- Isolate services
- Configure a secure file system
- Disable NetBIOS and server message block

Enhance Physical Security

Physical and logical isolation make up the foundation of SQL Server security. To enhance the physical security of the SQL Server installation, do the following tasks:

- Place the server in a room accessible only to authorized persons.
- Place computers that host a database in a physically protected location, ideally a locked computer room with monitored flood detection and fire detection or suppression systems.
- Install databases in the secure zone of the corporate intranet and do not connect your SQL Servers directly to the Internet.
- Back up all data regularly and secure the backups in an off-site location.

Use Firewalls

Firewalls are important to help secure the SQL Server installation. Firewalls will be most effective if you follow these guidelines:

- Put a firewall between the server and the Internet. Enable your firewall. If your firewall is turned off, turn it on. If your firewall is turned on, do not turn it off.

Notes

- Divide the network into security zones separated by firewalls. Block all traffic, and then selectively admit only what is required.
- In a multi-tier environment, use multiple firewalls to create screened subnets.
- When you are installing the server inside a Windows domain, configure interior firewalls to allow Windows Authentication.
- If your application uses distributed transactions, you might have to configure the firewall to allow Microsoft Distributed Transaction Co-ordinator (MS DTC) traffic to flow between separate MS DTC instances. You will also have to configure the firewall to allow traffic to flow between the MS DTC and resource managers such as SQL Server.

Isolate Services

Isolating services reduces the risk that one compromised service could be used to compromise others. To isolate services, consider the following guidelines:

- Run separate SQL Server services under separate Windows accounts. Whenever possible, use separate, low-rights Windows or Local user accounts for each SQL Server service.

Configure a Secure File System

Using the correct file system increases security. For SQL Server installations, you should do the following tasks:

- Use the NTFS file system (NTFS). NTFS is the preferred file system for installations of SQL Server because it is more stable and recoverable than FAT file systems. NTFS also enables security options like file and directory Access Control Lists (ACLs) and Encrypting File System (EFS) file encryption. During installation, SQL Server will set appropriate ACLs on registry keys and files if it detects NTFS. These permissions should not be changed. Future releases of SQL Server might not support installation on computers with FAT file systems.

Notes If you use EFS, database files will be encrypted under the identity of the account running SQL Server. Only this account will be able to decrypt the files. If you must change the account that runs SQL Server, you should first decrypt the files under the old account and then re-encrypt them under the new account.

- Use a Redundant Array of Independent Disks (RAID) for critical data files.

Disable NetBIOS and Server Message Block

Servers in the perimeter network should have all unnecessary protocols disabled, including NetBIOS and server message block (SMB).

NetBIOS uses the following ports:

- UDP/137 (NetBIOS name service)
- UDP/138 (NetBIOS datagram service)
- TCP/139 (NetBIOS session service)

SMB uses the following ports:

- TCP/139
- TCP/445

Notes

Web servers and Domain Name System (DNS) servers do not require NetBIOS or SMB. On these servers, disable both protocols to reduce the threat of user enumeration.

2.5.2 After Installing SQL Server

After installation, you can enhance the security of the SQL Server installation by following these best practices regarding accounts and authentication modes:

Service accounts

- Run SQL Server services by using the lowest possible permissions.
- Associate SQL Server services with low privileged Windows local user accounts, or domain user accounts.

Authentication mode

- Require Windows Authentication for connections to SQL Server.
- Use Kerberos authentication.

Strong passwords

- Always assign a strong password to the sa account.
- Always enable password policy checking for password strength and expiration.
- Always use strong passwords for all SQL Server logins.

Self Assessment

State true or false:

6. FAT is the preferred file system for installations of SQL Server because it is more stable and recoverable than NTFS file systems.
7. If you use EFS, database files will be encrypted under the identity of the account running SQL Server.
8. NetBIOS uses the following ports: UDP/137 (NetBIOS name service) and UDP/138 (NetBIOS datagram service) only.
9. SMB uses the following ports: TCP/139 and TCP/445
10. Web servers and Domain Name System (DNS) servers do not require NetBIOS or SMB.

2.6 Summary

- The SQL Server Installation Wizard is Windows Installer-based. It provides a single feature tree to install all SQL Server components:
- There are various ways of installing SQL server.
- SQL Server Setup requires Microsoft Windows Installer 4.5 or a later version, and Microsoft Data Access Components (MDAC) 2.8 SP1 or a later version.
- Microsoft Internet Explorer 6 SP1 or a later version is required for all installations of SQL Server 2008.

Notes

- SQL Server 2008 graphical tools require VGA or higher resolution: at least 1,024×768 pixel resolution.
- Security is important for every product and every business. By following simple best practices, you can avoid many security vulnerabilities.

2.7 Keywords

Default Database: The database the user is connected to immediately after logging in to SQL Server.

SQL Server: Microsoft SQL Server is a database management and analysis system for e-commerce, line-of-business, and data warehousing solutions.

SQL Server Installation Wizard: The SQL Server Installation Wizard is Windows Installer-based that provides a single feature tree to install all SQL Server components.

2.8 Review Questions

1. Describe the procedure of installing SQL server.
2. State the hardware and software requirements for installing SQL server 2008.
3. Explain the process of installing the Client Tools of Microsoft SQL Server.
4. Discuss the various security concerns associated with sql server installation.

Answers: Self Assessment

1.	Windows	2.	Various
3.	4.5	4.	6 SP1
5.	VGA	6.	False
7.	True	8.	False
9.	True	10.	True

2.9 Further Readings

Books

C.J. Date, *Introduction to Database Systems*, Pearson Education.

Elmasri Navrate, *Fundamentals of Database Systems*, Pearson Education.

Peter Rob & Carlos Coronel, *Database Systems Design, Implementation and Management*, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, *Database Management Systems*, Tata McGraw Hill, 3rd Edition.

Silberschatz, Korth, *Database System Concepts*, McGraw Hill, 5th Edition.

Online links

www.sql-tutorial.net/

www.w3schools.com/sql/

www.sqlservertutorials.com/

Notes

Unit 3: SQL Server Tools

CONTENTS

Objectives

Introduction

3.1 Management Studio

3.1.1 To Open SQL Server Management Studio

3.1.2 Management Studio Components

3.1.3 Showing Additional Windows

3.2 Log File Viewer

3.2.1 Opening Log File Viewer

3.2.2 Permissions

3.3 Summary

3.4 Keywords

3.5 Review Questions

3.6 Further Readings

Objectives

After studying this unit, you will be able to:

- Learn about SQL server tools
- Understand the meaning and usage of management studio
- Know about various components of management studio
- Discuss the concept of Log File Viewer
- Comprehend how Log File Viewer can be used to view various types of information

Introduction

Microsoft SQL Server includes a complete set of graphical tools and command line utilities that allow users, programmers, and administrators to increase their productivity. This unit will help you learn to get the most out of SQL Server tools so you can work efficiently, right from the start.

3.1 Management Studio

Management Studio is a new tool for SQL Server database administrators and developers. Hosted inside Microsoft Visual Studio, it brings graphical tools for database management together with a rich development environment. Management Studio includes the functions of SQL Server 2000 Enterprise Manager, Analysis Manager, and SQL Query Analyzer in one tool, along with the ability to write MDX, XMLA, and XML statements.

This unit will help you understand the presentation of information in Management Studio and how to take advantage of the features that were not available in Enterprise Manager.

Notes

3.1.1 To Open SQL Server Management Studio

1. On the **Start** menu, point to **All Programs**, point to **Microsoft SQL Server 2005**, and then click **SQL Server Management Studio**.
2. In the **Connect to Server** dialog box, verify the default settings, and then click **Connect**. To connect, the **Server name** box must contain the name of the computer where SQL Server is installed. If the Database Engine is a named instance, the Server name box should also contain the instance name in the format *<computer_name>\<instance_name>*.

3.1.2 Management Studio Components

Management Studio presents information in windows dedicated to specific types of information. Database information is shown in Object Explorer and document windows.

Tool	Purpose
Using Object Explorer	Browse servers, create and locate objects, manage data sources, and view logs. This tool is accessed from the **View** menu.
Managing Registered Servers	Store connection information for frequently accessed servers. This tool is accessed from the **View** menu.
Using Solution Explorer	Store and organize scripts and related connection information in projects called SQL Server Scripts. You can store several SQL Server Scripts as Solutions and use source control to manage scripts as they evolve over time. This tool is accessed from the **View** menu.
Using SQL Server Management Studio Templates	Create queries based on existing templates. You can also create your custom queries or alter the existing templates to fit your scenarios. This tool is accessed from the **View** menu.
Dynamic Help	Show a list of related Help topics as you click on a component or type code.

The tools in SQL Server Management Studio work together. For example, you can:

- Register a server with Object Explorer.
- Open a SQL Editor window connected to a specific database from Object Explorer.
- Object Explorer is a tree view of all the database objects in a server. This can include the databases of the SQL Server Database Engine, Analysis Services, Reporting Services, Integration Services, and SQL Server 2005 Compact Edition. Object Explorer includes information for all servers to which it is connected. When you open Management Studio, you are prompted to connect Object Explorer to the settings that were last used. You can double-click any server in the Registered Servers component to connect to it, but you do not have to register a server to connect.
- The document window is the largest portion of Management Studio. The document windows can contain query editors and browser windows. By default, the Object Explorer Details page is displayed, connected to the instance of Database Engine on the current computer.

Task List the various components of Management Studio.

Notes

3.1.3 Showing Additional Windows

Notes Users familiar with the SQL Server Enterprise Manager of SQL Server 2000 may wish to show the Registered Servers window.

To show the Registered Servers window

1. On the View menu, **click Registered Servers**.

   The Registered Servers window appears above Object Explorer. Registered Servers lists servers which you manage frequently. You can add and remove servers from this list. If SQL Server 2000 Enterprise Manager was previously installed on this computer, you will be prompted to import the list of registered servers. Otherwise, the only servers listed are the SQL Server instances on the computer where you are running Management Studio.

2. If your server does not appear, in Registered Servers, right-click Database Engine, and then click Update Local Server Registration.

Self Assessment

Fill in the blanks:

1. Database information is shown in …….. and document windows.
2. The …….. is the largest portion of Management Studio.
3. …….. is a new tool for SQL Server database administrators and developers.
4. Object Explorer includes information for all …….. to which it is connected.
5. The Registered Servers window appears above ……..
6. Registered Servers lists servers which you manage ……..
7. By default, the Object Explorer Details page is displayed, connected to the instance of …….. on the current computer.

3.2 Log File Viewer

Log File Viewer in SQL Server Management Studio is used to access information about errors and events that are captured in the following logs:

- Audit Collection
- Data Collection
- Database Mail
- Job History
- Maintenance Plans
- Remote Maintenance Plans
- SQL Server
- SQL Server Agent
- Windows NT (These are Windows events that can also be accessed from Event Viewer.)

Notes

3.2.1 Opening Log File Viewer

How to Open Log File Viewer.

Did u know? You can open Log File Viewer in several ways, depending on the information that you want to view.

To view logs that are related to general SQL Server activity

1. In Object Explorer, expand Management.
2. Do either of the following:
   i. Right-click SQL Server Logs, point to View, and then click either SQL Server Log or SQL Server and Windows Log.
   ii. Expand SQL Server Logs, right-click any log file, and then click View SQL Server Log. You can also double-click any log file.

   The logs include Database Mail, SQL Server, SQL Server Agent, and Windows NT.

To view logs that are related to jobs

- To view job history, in Object Explorer, expand SQL Server Agent, right-click Jobs, and then click View History.

  The logs include Database Mail, Job History, and SQL Server Agent.
- Another way to view the SQL Server Agent log is to expand SQL Server Agent, expand Error Logs, right-click a log file, and then click View Agent Log.

  The logs include Database Mail, SQL Server, SQL Server Agent, and Windows NT.

To view logs that are related to maintenance plans

- In Object Explorer, expand Management, right-click Maintenance Plans, and then click View History.

  The logs include Database Mail, Job History, Maintenance Plans, Remote Maintenance Plans, and SQL Server Agent.

  To view logs that are related to Data Collection.
- In Object Explorer, expand Management, right-click Data Collection, and then click View Logs.

  The logs include Data Collection, Job History, and SQL Server Agent.

To view logs that are related to Database Mail

- In Object Explorer, expand Management, right-click Database Mail, and then click View Database Mail Log.

  The logs include Database Mail, Job History, Maintenance Plans, Remote Maintenance Plans, SQL Server, SQL Server Agent, and Windows NT.

To view logs that are related to audits collections

- In Object Explorer, expand Security, expand Audits, right-click an audit, and then click View Audit Logs.

  The logs include Audit Collection and Windows NT.

Notes

3.2.2 Permissions

Requires membership in the securityadmin fixed server role.

Self Assessment

State true or false:

8. You cannot open Log File Viewer in several ways, depending on the information that you want to view.
9. Log File Viewer in SQL Server Management Studio is used to access information about errors and events.
10. Remote Maintenance Plans are Windows events that can also be accessed from Event Viewer apart from Log File Viewer.

3.3 Summary

- Management Studio includes the functions of SQL Server 2000 Enterprise Manager, Analysis Manager, and SQL Query Analyzer in one tool, along with the ability to write MDX, XMLA, and XML statements.
- Management Studio presents information in windows dedicated to specific types of information. Database information is shown in Object Explorer and document windows.
- The tools in SQL Server Management Studio work together.
- Object Explorer is a tree view of all the database objects in a server. This can include the databases of the SQL Server Database Engine, Analysis Services, Reporting Services, Integration Services, and SQL Server 2005 Compact Edition.
- The document window is the largest portion of Management Studio. The document windows can contain query editors and browser windows.
- **Log File Viewer** in SQL Server Management Studio is used to access information about errors and events. You can open Log File Viewer in several ways, depending on the information that you want to view.

3.4 Keywords

Log File Viewer: Log File Viewer in SQL Server Management Studio is used to access information about errors and events.

Management Studio: Management Studio presents information in windows dedicated to specific types of information.

Object Explorer: Object Explorer is a tree view of all the database objects in a server.

3.5 Review Questions

1. Explain the use of SQL server tools.
2. Briefly describe various types of sql server tools.
3. Explain the concept of "Management Studio" in SQL server.

Notes

4. Discuss the various components of management studio.
5. What is Log File viewer in SQL server? Also discuss its usage.
6. Analyze the statement: "You can open Log File Viewer in several ways, depending on the information that you want to view."
7. Explain how to open Log File Viewer:
   (a) To view logs that are related to general SQL Server activity
   (b) To view logs that are related to jobs
   (c) To view logs that are related to maintenance plans
8. Describe how you will show the Registered Servers window in SQL server enterprise manager.

Answers: Self Assessment

1.	Object Explorer	2.	document window
3.	Management Studio	4.	Servers
5.	Object Explorer	6.	Frequently
7.	Database Engine	8.	False
9.	True	10.	False

3.6 Further Readings

Books

C.J. Date, *Introduction to Database Systems*, Pearson Education.

Elmasri Navrate, *Fundamentals of Database Systems*, Pearson Education.

Peter Rob & Carlos Coronel, *Database Systems Design, Implementation and Management*, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, *Database Management Systems*, Tata McGraw Hill, 3rd Edition.

Silberschatz, Korth, *Database System Concepts*, McGraw Hill, 5th Edition.

Online links

www.sql-tutorial.net/

www.w3schools.com/sql/

www.sqlservertutorials.com/

Notes

Unit 4: SQL Server Storage Architecture

CONTENTS

Objectives

Introduction

4.1 Resource Database

4.1.1 Physical Properties of Resource

4.1.2 Backing Up and Restoring the Resource Database

4.1.3 Accessing the Resource Database

4.2 Database Physical Structure

4.2.1 Pages

4.2.2 Extents

4.3 Database Files

4.3.1 Logical and Physical File Names

4.3.2 Data File Pages

4.3.3 File Size

4.3.4 Database Snapshot Files

4.4 Transaction Log

4.4.1 Operations Supported by the Transaction Log

4.4.2 Transaction Log Characteristics

4.5 Summary

4.6 Keywords

4.7 Review Questions

4.8 Further Readings

Objectives

After studying this unit, you will be able to:

- Discuss the meaning and importance of resource database
- Describe how SQL Server databases and files are organized.
- Comprehend the use of database files
- Understand the concept of transaction log and its importance in SQL server

Introduction

This unit will explore the concepts relating to SQL Server storage architecture. It will help you understand the meaning of resource database and how to work with resource database.

Notes

It will present some benefits and concerns about different storage designs so that you can more easily choose what suits your solution. We will finish the chapter with a presentation of SQL Server-examining its architecture and how it fits into your storage policy.

4.1 Resource Database

The Resource database is a read-only database that contains all the system objects that are included with SQL Server. SQL Server system objects, such as sys.objects, are physically persisted in the Resource database, but they logically appear in the sys schema of every database. The Resource database does not contain user data or user metadata.

The Resource database makes upgrading to a new version of SQL Server an easier and faster procedure. In earlier versions of SQL Server, upgrading required dropping and creating system objects. Because the Resource database file contains all system objects, an upgrade is now accomplished simply by copying the single Resource database file to the local server.

4.1.1 Physical Properties of Resource

The physical file names of the Resource database are mssqlsystemresource.mdf and mssqlsystemresource.ldf. These files are located in <drive>:\Program Files\Microsoft SQL Server\MSSQL10_50.<instance_name>\MSSQL\Binn\. Each instance of SQL Server has one and only one associated mssqlsystemresource.mdf file, and instances do not share this file.

4.1.2 Backing Up and Restoring the Resource Database

SQL Server cannot back up the Resource database. You can perform your own file-based or a disk-based backup by treating the mssqlsystemresource.mdf file as if it were a binary (.EXE) file, rather than a database file, but you cannot use SQL Server to restore your backups. Restoring a backup copy of mssqlsystemresource.mdf can only be done manually, and you must be careful not to overwrite the current Resource database with an out-of-date or potentially insecure version.

Notes After restoring a backup of mssqlsystemresource.mdf, you must reapply any subsequent updates.

4.1.3 Accessing the Resource Database

The Resource database should only be modified by or at the direction of a Microsoft Customer Support Services (CSS) specialist. The ID of the Resource database is always 32767. Other important values associated with the Resource database are the version number and the last time that the database was updated.

To determine the version number of the Resource database, use:

```
SELECT SERVERPROPERTY('ResourceVersion');
GO
```

To determine when the Resource database was last updated, use:

```
SELECT SERVERPROPERTY('ResourceLastUpdateDateTime');
GO
```

Notes

To access SQL definitions of system objects, use the OBJECT_DEFINITION function:

SELECT OBJECT_DEFINITION(OBJECT_ID('sys.objects'));

GO

4.2 Database Physical Structure

The topics in this section describe how SQL Server databases and files are organized.

The fundamental unit of data storage in SQL Server is the page. The disk space allocated to a data file (.mdf or .ndf) in a database is logically divided into pages numbered contiguously from 0 to *n*. Disk I/O operations are performed at the page level. That is, SQL Server reads or writes whole data pages.

Extents are a collection of eight physically contiguous pages and are used to efficiently manage the pages. All pages are stored in extents.

4.2.1 Pages

In SQL Server, the page size is 8 KB. This means SQL Server databases have 128 pages per megabyte. Each page begins with a 96-byte header that is used to store system information about the page. This information includes the page number, page type, the amount of free space on the page, and the allocation unit ID of the object that owns the page.

The following table shows the page types used in the data files of a SQL Server database.

Table 4.1

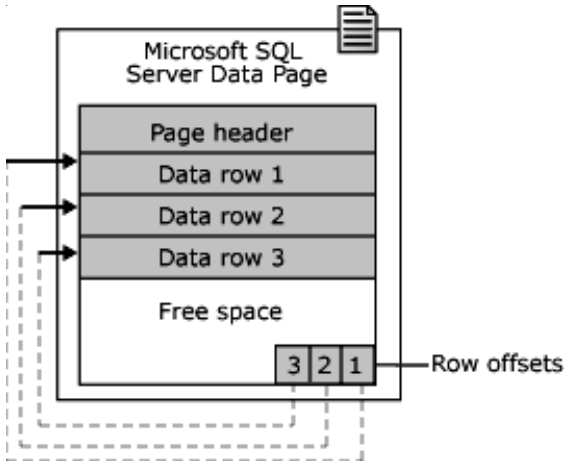
Page type	Contents
Data	Data rows with all data, except **text**, **ntext**, **image**, **nvarchar(max)**, **varchar(max)**, **varbinary(max)**, and **xml** data, when **text in row** is set to ON.
Index	Index entries.
Text/Image	Large object data types: **text**, **ntext**, **image**, **nvarchar(max)**, **varchar(max)**, **varbinary(max)**, and **xml** data Variable length columns when the data row exceeds 8 KB: **varchar**, **nvarchar**, **varbinary**, and **sql_variant**
Global Allocation Map, Shared Global Allocation Map	Information about whether extents are allocated.
Page Free Space	Information about page allocation and free space available on pages.
Index Allocation Map	Information about extents used by a table or index per allocation unit.
Bulk Changed Map	Information about extents modified by bulk operations since the last BACKUP LOG statement per allocation unit.
Differential Changed Map	Information about extents that have changed since the last BACKUP DATABASE statement per allocation unit.

Notes Log files do not contain pages; they contain a series of log records.

Notes

Data rows are put on the page serially, starting immediately after the header. A row offset table starts at the end of the page, and each row offset table contains one entry for each row on the page. Each entry records how far the first byte of the row is from the start of the page. The entries in the row offset table are in reverse sequence from the sequence of the rows on the page.

Figure 4.1



Large Row Support

Rows cannot span pages in SQL Server 2005, however portions of the row may be moved off the row's page so that the row can actually be very large. The maximum amount of data and overhead that is contained in a single row on a page is 8,060 bytes (8 KB). However, this does not include the data stored in the Text/Image page type. In SQL Server 2005, this restriction is relaxed for tables that contain **varchar**, **nvarchar**, **varbinary**, or **sql_variant** columns. When the total row size of all fixed and variable columns in a table exceeds the 8,060 byte limitation, SQL Server dynamically moves one or more variable length columns to pages in the ROW_OVERFLOW_DATA allocation unit, starting with the column with the largest width. This is done whenever an insert or update operation increases the total size of the row beyond the 8060 byte limit. When a column is moved to a page in the ROW_OVERFLOW_DATA allocation unit, a 24-byte pointer on the original page in the IN_ROW_DATA allocation unit is maintained. If a subsequent operation reduces the row size, SQL Server dynamically moves the columns back to the original data page.

4.2.2 Extents

Extents are the basic unit in which space is managed. An extent is eight physically contiguous pages, or 64 KB. This means SQL Server databases have 16 extents per megabyte.

To make its space allocation efficient, SQL Server does not allocate whole extents to tables with small amounts of data. SQL Server has two types of extents:

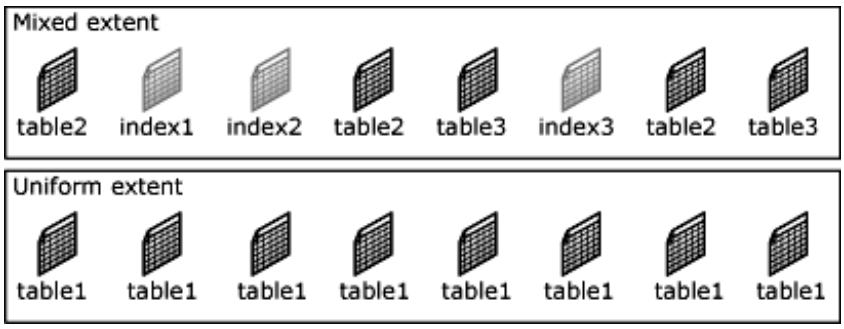
- Uniform extents are owned by a single object; all eight pages in the extent can only be used by the owning object.
- Mixed extents are shared by up to eight objects. Each of the eight pages in the extent can be owned by a different object.

A new table or index is generally allocated pages from mixed extents. When the table or index grows to the point that it has eight pages, it then switches to use uniform extents for subsequent

Notes

allocations. If you create an index on an existing table that has enough rows to generate eight pages in the index, all allocations to the index are in uniform extents.

Figure 4.2



Self Assessment

State true or false:

1. All extents are stored in pages.
2. The fundamental unit of data storage in SQL Server is the page.
3. Data rows are put on the page serially, starting immediately after the header.
4. The minimum amount of data and overhead that is contained in a single row on a page is 8,060 bytes (8 KB).
5. Each instance of SQL Server has one and only one associated mssqlsystemresource.mdf file, and instances are allowed to share this file.
6. An extent is eight physically contiguous pages, or 64 KB.

4.3 Database Files

SQL Server 2005 maps a database over a set of operating-system files. Data and log information are never mixed in the same file, and individual files are used only by one database. Filegroups are named collections of files and are used to help with data placement and administrative tasks such as backup and restore operations.

SQL Server 2005 databases have three types of files:

- ***Primary data files:*** The primary data file is the starting point of the database and points to the other files in the database. Every database has one primary data file. The recommended file name extension for primary data files is .mdf.
- ***Secondary data files:*** Secondary data files make up all the data files, other than the primary data file. Some databases may not have any secondary data files, while others have several secondary data files. The recommended file name extension for secondary data files is .ndf.
- ***Log files:*** Log files hold all the log information that is used to recover the database. There must be at least one log file for each database, although there can be more than one. The recommended file name extension for log files is .ldf.

SQL Server 2005 does not enforce the .mdf, .ndf, and .ldf file name extensions, but these extensions help you identify the different kinds of files and their use.

Notes

In SQL Server 2005, the locations of all the files in a database are recorded in the primary file of the database and in the master database. The SQL Server Database Engine uses the file location information from the **master** database most of the time. However, the Database Engine uses the file location information from the primary file to initialize the file location entries in the **master** database in the following situations:

- When attaching a database using the CREATE DATABASE statement with either the FOR ATTACH or FOR ATTACH_REBUILD_LOG options.
- When upgrading from SQL Server version 2000 or version 7.0 to SQL Server 2005.
- When restoring the **master** database.

4.3.1 Logical and Physical File Names

SQL Server files have two names:

logical_file_name

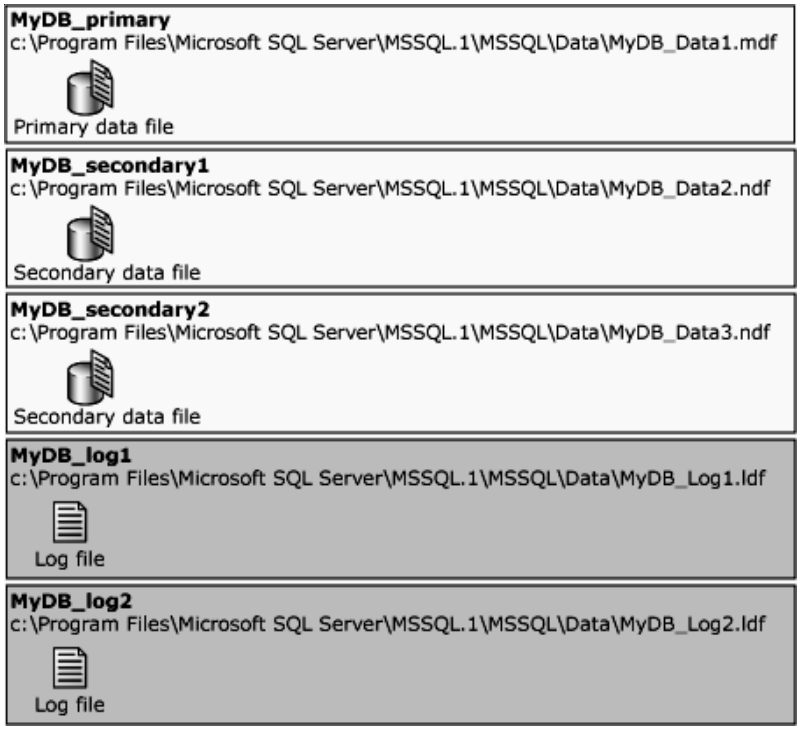
The *logical_file_name* is the name used to refer to the physical file in all Transact-SQL statements. The logical file name must comply with the rules for SQL Server identifiers and must be unique among logical file names in the database.

os_file_name

The *os_file_name* is the name of the physical file including the directory path. It must follow the rules for the operating system file names.

The following illustration shows examples of the logical file names and the physical file names of a database created on a default instance of SQL Server 2005:

Figure 4.3



Notes

SQL Server data and log files can be put on either FAT or NTFS file systems. We recommend using the NTFS file system because the security aspects of NTFS. Read/write data filegroups and log files cannot be placed on an NTFS compressed file system. Only read-only databases and read-only secondary filegroups can be put on an NTFS compressed file system.

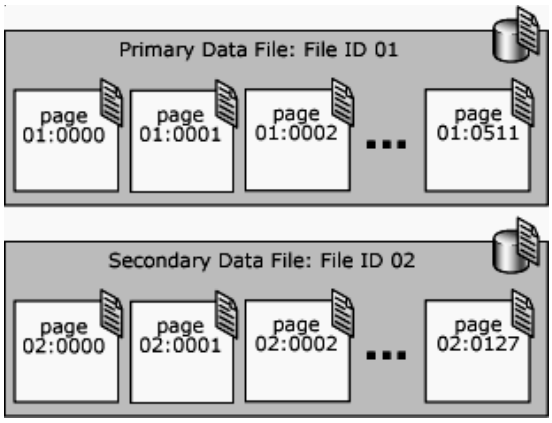
Task Differentiate between *logical_file_name and os_file_name*

When multiple instances of SQL Server are run on a single computer, each instance receives a different default directory to hold the files for the databases created in the instance.

4.3.2 Data File Pages

Pages in a SQL Server data file are numbered sequentially, starting with zero (0) for the first page in the file. Each file in a database has a unique file ID number. To uniquely identify a page in a database, both the file ID and the page number are required. The following example shows the page numbers in a database that has a 4-MB primary data file and a 1-MB secondary data file.

Figure 4.4



The first page in each file is a file header page that contains information about the attributes of the file. Several of the other pages at the start of the file also contain system information, such as allocation maps. One of the system pages stored in both the primary data file and the first log file is a database boot page that contains information about the attributes of the database.

4.3.3 File Size

SQL Server 2005 files can grow automatically from their originally specified size. When you define a file, you can specify a specific growth increment. Every time the file is filled, it increases its size by the growth increment. If there are multiple files in a filegroup, they will not autogrow until all the files are full. Growth then occurs in a round-robin fashion.

Each file can also have a maximum size specified. If a maximum size is not specified, the file can continue to grow until it has used all available space on the disk. This feature is especially useful when SQL Server is used as a database embedded in an application where the user does not have convenient access to a system administrator. The user can let the files autogrow as required to reduce the administrative burden of monitoring free space in the database and manually allocating additional space.

Notes

4.3.4 Database Snapshot Files

The form of file that is used by a database snapshot to store its copy-on-write data depends on whether the snapshot is created by a user or used internally:

- A database snapshot that is created by a user stores its data in one or more sparse files. Sparse file technology is a feature of the NTFS file system. At first, a sparse file contains no user data, and disk space for user data has not been allocated to the sparse file.
- Database snapshots are used internally by certain DBCC commands. These commands include DBCC CHECKDB, DBCC CHECKTABLE, DBCC CHECKALLOC, and DBCC CHECKFILEGROUP. An internal database snapshot uses sparse alternate data streams of the original database files.

Notes Like sparse files, alternate data streams are a feature of the NTFS file system. The use of sparse alternate data streams allows for multiple data allocations to be associated with a single file or folder without affecting the file size or volume statistics.

4.4 Transaction Log

Every SQL Server 2005 database has a transaction log that records all transactions and the database modifications made by each transaction. The transaction log is a critical component of the database and, if there is a system failure, the transaction log might be required to bring your database back to a consistent state. The transaction log should never be deleted or moved unless you fully understand the ramifications of doing this.

4.4.1 Operations Supported by the Transaction Log

The transaction log supports the following operations:

- Recovery of individual transactions.
- Recovery of all incomplete transactions when SQL Server is started.
- Rolling a restored database, file, filegroup, or page forward to the point of failure.
- Supporting transactional replication.
- Supporting standby-server solutions.

Recovery of Individual Transactions

If an application issues a ROLLBACK statement, or if the Database Engine detects an error such as the loss of communication with a client, the log records are used to roll back the modifications made by an incomplete transaction.

Recovery of all Incomplete Transactions when SQL Server is Started

If a server that is running SQL Server fails, the databases may be left in a state where some modifications were never written from the buffer cache to the data files, and there may be some modifications from incomplete transactions in the data files. When an instance of SQL Server is started, it runs a recovery of each database. Every modification recorded in the log which may not have been written to the data files is rolled forward. Every incomplete transaction found in the transaction log is then rolled back to make sure the integrity of the database is preserved.

Notes

Rolling a Restored Database, File, Filegroup, or Page Forward to the Point of Failure

After a hardware loss or disk failure affecting the database files, you can restore the database to the point of failure. You first restore the last full database backup and the last differential database backup, and then restore the subsequent sequence of the transaction log backups to the point of failure. As you restore each log backup, the Database Engine reapplies all the modifications recorded in the log to roll forward all the transactions. When the last log backup is restored, the Database Engine then uses the log information to roll back all transactions that were not complete at that point.

Supporting Transactional Replication

The Log Reader Agent monitors the transaction log of each database configured for transactional replication and copies the transactions marked for replication from the transaction log into the distribution database.

Supporting Standby-Server Solutions

The standby-server solutions, database mirroring, and log shipping, rely heavily on the transaction log. In a log shipping scenario, the primary server sends the active transaction log of the primary database to one or more destinations. Each secondary server restores the log to its local secondary database.

In a database mirroring scenario, every update to a database, the principal database, is immediately reproduced in a separate, full copy of the database, the mirror database. The principal server instance sends each log record immediately to the mirror server instance which applies the incoming log records to the mirror database, continually rolling it forward.

4.4.2 Transaction Log Characteristics

Following are the characteristics of the SQL Server Database Engine transaction log:

- The transaction log is implemented as a separate file or set of files in the database. The log cache is managed separately from the buffer cache for data pages, which results in simple, fast, and robust code within the Database Engine.
- The format of log records and pages is not constrained to follow the format of data pages.
- The transaction log can be implemented in several files. The files can be defined to expand automatically by setting the FILEGROWTH value for the log. This reduces the potential of running out of space in the transaction log, while at the same time reducing administrative overhead.
- The mechanism to reuse the space within the log files is quick and has minimal effect on transaction throughput.

Self Assessment

Fill in the blanks:

7. …… are named collections of files and are used to help with data placement and administrative tasks such as backup and restore operations.
8. …… is the name used to refer to the physical file in all Transact-SQL statements.
9. …… is the name of the physical file including the directory path.

Notes

10. Every SQL Server 2005 database has a …… that records all transactions and the database modifications made by each transaction.

4.5 Summary

- The Resource database is a read-only database that contains all the system objects that are included with SQL Server. SQL Server system objects, such as sys.objects, are physically persisted in the Resource database, but they logically appear in the sys schema of every database. The Resource database does not contain user data or user metadata.
- The physical file names of the Resource database are mssqlsystemresource.mdf and mssqlsystemresource.ldf.
- SQL Server cannot back up the Resource database.
- The Resource database should only be modified by or at the direction of a Microsoft Customer Support Services (CSS) specialist. The ID of the Resource database is always 32767.
- In SQL Server, the page size is 8 KB. This means SQL Server databases have 128 pages per megabyte. Each page begins with a 96-byte header that is used to store system information about the page.
- Extents are the basic unit in which space is managed. An extent is eight physically contiguous pages, or 64 KB. This means SQL Server databases have 16 extents per megabyte.
- Filegroups are named collections of files and are used to help with data placement and administrative tasks such as backup and restore operations.
- SQL Server 2005 databases have three types of files: Primary, secondary and Log files.
- The *logical_file_name* is the name used to refer to the physical file in all Transact-SQL statements.
- The *os_file_name* is the name of the physical file including the directory path.
- Every SQL Server 2005 database has a transaction log that records all transactions and the database modifications made by each transaction.

4.6 Keywords

Extent: Extents are a collection of eight physically contiguous pages and are used to efficiently manage the pages.

File Groups: File groups are named collections of files and are used to help with data placement and administrative tasks such as backup and restore operations.

***logical_file_name*:** The *logical_file_name* is the name used to refer to the physical file in all Transact-SQL statements.

os_file_name: The *os_file_name* is the name of the physical file including the directory path.

Page: The fundamental unit of data storage in SQL Server is the page

Resource Database: The Resource database is a read-only database that contains all the system objects that are included with SQL Server.

Transaction Log: Every SQL Server 2005 database has a transaction log that records all transactions and the database modifications made by each transaction.

Notes

4.7 Review Questions

1. What do you understand by the term: Resource Database?
2. Describe the database physical structure.
3. Explain database files.
4. Discuss the use of transaction log in SQL server.
5. Describe the characteristics of the SQL Server Database Engine transaction log.
6. Differentiate between a Page and a Extent.
7. Briefly describe the various types of page sin SQL server.
8. Explain the various types of file types in SQL Databases.
9. Write a short note on: Logical and Physical File Names in SQL Server.

Answers: Self Assessment

1.	False	2.	True
3.	True	4.	False
5.	False	6.	True
7.	File groups	8.	*logical_file_name*
9.	*os_file_name*	10.	transaction log

4.8 Further Readings

Books

C.J. Date, *Introduction to Database Systems*, Pearson Education.

Elmasri Navrate, *Fundamentals of Database Systems*, Pearson Education.

Peter Rob & Carlos Coronel, *Database Systems Design, Implementation and Management*, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, *Database Management Systems*, Tata McGraw Hill, 3rd Edition.

Silberschatz, Korth, *Database System Concepts*, McGraw Hill, 5th Edition.

Online links

www.sql-tutorial.net/

www.w3schools.com/sql/

www.sqlservertutorials.com/

Unit 5: SQL Server Databases

Notes

CONTENTS

Objectives
Introduction
5.1 System Databases
5.1.1 Modifying System Data
5.1.2 Viewing System Database Data
5.2 User Databases
5.3 Database Planning
5.4 Creating Databases
5.5 Tables
5.5.1 Designing Tables
5.5.2 Creating and Modifying a Table
5.6 Constraints
5.6.1 Classes of Constraints
5.6.2 Column and Table Constraints
5.7 Database Diagrams
5.7.1 Creating Database Diagrams
5.8 Views
5.9 Synonyms
5.9.1 Syntax for Creating Synonym
5.9.2 Arguments
5.9.3 Permissions
5.10 Programming Objects
5.10.1 Creating Procedures
5.10.2 Functions
5.10.3 Triggers
5.11 Summary
5.12 Keywords
5.13 Review Questions
5.14 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand meaning of databases
- Describes the system databases used in SQL Server

Notes

- Learn database planning essentials
- Describes the purpose of the database and how it affects the design
- Provides guidelines for creating a database plan to fit your purpose
- Explains the concept of synonyms and the benefits of using synonyms

Introduction

A database in SQL Server is made up of a collection of tables. These tables contain data and other objects, such as views, indexes, stored procedures, user-defined functions, and triggers that are defined to support activities performed with the data. The data stored in a database is typically related to a particular subject or process, such as inventory information for a manufacturing warehouse.

5.1 System Databases

SQL Server includes the following system databases.

Table 5.1

System database	Description
master Database	Records all the system-level information for an instance of SQL Server.
msdb Database	Is used by SQL Server Agent for scheduling alerts and jobs.
model Database	Is used as the template for all databases created on the instance of SQL Server. Modifications made to the **model** database, such as database size, collation, recovery model, and other database options, are applied to any databases created afterward.
Resource Database	Is a read-only database that contains system objects that are included with SQL Server. System objects are physically persisted in the **Resource** database, but they logically appear in the **sys** schema of every database.
tempdb Database	Is a workspace for holding temporary objects or intermediate result sets.

5.1.1 Modifying System Data

SQL Server does not support users directly updating the information in system objects such as system tables, system stored procedures, and catalog views. Instead, SQL Server provides a complete set of administrative tools that let users fully administer their system and manage all users and objects in a database. These include the following:

- ***Administration utilities,*** such as SQL Server Management Studio.
- ***SQL-SMO API.*** This lets programmers include complete functionality for administering SQL Server in their applications.
- ***Transact-SQL scripts and stored procedures.*** These can use system stored procedures and Transact-SQL DDL statements.

These tools shield applications from changes in the system objects. For example, SQL Server sometimes has to change the system tables in new versions of SQL Server to support new functionality that is being added in that version. Applications issuing SELECT statements that directly reference system tables are frequently dependent on the old format of the system tables. Sites may not be able to upgrade to a new version of SQL Server until they have rewritten applications that are selecting from system tables. SQL Server considers the system stored

procedures, DDL, and SQL-SMO published interfaces, and works to maintain the backward compatibility of these interfaces.

Did u know? SQL Server does not support triggers defined on the system tables, because they might modify the operation of the system.

Notes System databases cannot reside on UNC share directories.

5.1.2 Viewing System Database Data

You should not code Transact-SQL statements that directly query the system tables, unless that is the only way to obtain the information that is required by the application. Instead, applications should obtain catalog and system information by using the following:

- System catalog views
- SQL-SMO
- Windows Management Instrumentation (WMI) interface
- Catalog functions, methods, attributes, or properties of the data API used in the application, such as ADO, OLE DB, or ODBC.
- Transact-SQL system stored procedures and built-in functions.

5.2 User Databases

In SQL Server, you can move the data, log, and full-text catalog files of a user database to a new location by specifying the new file location in the FILENAME clause of the ALTER DATABASE statement. This method applies to moving database files within the same instance SQL Server. To move a database to another instance of SQL Server or to another server, use backup and restore or detach and attach operations.

Notes Some features of the SQL Server Database Engine change the way that the Database Engine stores information in the database files. These features are restricted to specific editions of SQL Server. A database that contains these features cannot be moved to an edition of SQL Server that does not support them. Use the sys.dm_db_persisted_sku_features dynamic management view to list all edition-specific features that are enabled in the current database.

The procedures in this topic require the logical name of the database files. To obtain the name, query the name column in the sys.master_files catalog view.

When you move a database onto another server instance, to provide a consistent experience to users and applications, you might have to recreate some or all the metadata for the database.

5.3 Database Planning

Designing a database requires an understanding of the business functions you want to model. It also requires an understanding of the database concepts and features that you want to use to

Notes

represent those business functions. Make sure that you accurately design the database to model the business, because it can be time-consuming to significantly change the design of a database after you implement it. A well-designed database also performs better.

The first step in creating a database is creating a plan that serves both as a guide to be used when implementing the database and as a functional specification for the database after it has been implemented. The complexity and detail of a database design is dictated by the complexity and size of the database application and also the user population.

The nature and complexity of a database application, and also the process of planning it, can vary significantly. A database can be relatively simple and designed for use by a single person, or it can be large and complex and designed, for example, to handle all the banking transactions for thousands of clients. In the first case, the database design may be slightly more than a few notes on some scratch paper. In the latter case, the design may be a formal document hundreds of pages long that contains every possible detail about the database.

In planning the database, regardless of its size and complexity, use the following basic steps:

- Gather information.
- Identify the objects.
- Model the objects.
- Identify the types of information for each object.
- Identify the relationships between objects.

Step 1: Gathering Information

Before creating a database, you must have a good understanding of the job the database is expected to perform. If the database is to replace a paper-based or manually performed information system, the existing system will give you most of the information that you need. You should interview everyone involved in the system to determine what they do and what they need from the database. It is also important to identify what they want the new system to do, and also to identify the problems, limitations, and bottlenecks of any existing system. Collect copies of customer statements, inventory lists, management reports, and any other documents that are part of the existing system, because these will be useful to you in designing the database and the interfaces.

Step 2: Identifying the Objects

During the process of gathering information, you must identify the key objects or entities that will be managed by the database. The object can be a tangible thing, such as a person or a product, or it can be a more intangible item, such as a business transaction, a department in a company, or a payroll period. There are generally a few primary objects, and after these are identified, the related items become visible. Each distinct item in your database should have a corresponding table.

The primary object in the AdventureWorks2008R2 sample database included with SQL Server is a bicycle. The objects related to bicycle within this company's business are the employees who manufacture the bicycle, the vendors that sell components used to manufacture the bicycle, the customers who buy them, and the sales transactions performed with the customers. Each of these objects is a table in the database.

Step 3: Modeling the Objects

As the objects in the system are identified, you should record them in a way that represents the system visually. You can use your database model as a reference during implementation of the database.

Notes

For this purpose, database developers use tools that range in technical complexity from pencils and scratch paper to word processing and spreadsheet programs, and even to software programs created specifically for the job of data modeling for database designs. Whatever tool you decide to use, it is important that you keep it up to date.

Step 4: Identifying the Types of Information for Each Object

After the primary objects in the database have been identified as candidates for tables, the next step is to identify the types of information that must be stored for each object. These are the columns in the table of the object. The columns in a database table contain a few common types of information:

- ***Raw data columns:*** These columns store tangible pieces of information, such as names, determined by a source external to the database.
- ***Categorical columns:*** These columns classify or group the data and store a limited selection of data such as true/false, married/single, and VP/Director/Group Manager.
- ***Identifier columns:*** These columns provide a mechanism to identify each item stored in the table. These columns frequently have an ID or number in their name, for example, employee_id, invoice_number, and publisher_id. The identifier column is the primary component used by both users and internal database processing for gaining access to a row of data in the table. Sometimes the object has a tangible form of ID used in the table, for example, a social security number, but in most situations you can define the table so that a reliable, artificial ID can be created for the row.
- ***Relational or referential columns:*** These columns establish a link between information in one table and related information in another table. For example, a table that tracks sales transactions will generally have a link to the customers table so that the complete customer information can be associated with the sales transaction.

Step 5: Identifying the Relationship between Objects

One of the strengths of a relational database is the ability to relate or associate information about various items in the database. Isolated types of information can be stored separately, but the database engine can combine data when it is required. Identifying the relationship between objects in the design process requires looking at the tables, determining how they are logically related, and adding relational columns that establish a link from one table to another.

For example, the designer of the AdventureWorks2008R2 database has created tables for products and product models in the database. The Production.Product table contains information for each product that includes an identifier column named ProductID; data columns for the product name, the price of the product, and the product color, size, and weight. The table contains categorical columns, such as Class, or Style, that lets the products be grouped by these types. Each product also has a product model, but that information is stored in another table. Therefore, the Production.Product table has a ProductModelID column to store just the ID of the product model. When a row of data is added for a product, the value for ProductModelID must exist in the Production.ProductModel table.

5.4 Creating Databases

To create a database determine the name of the database, its owner (the user who creates the database), its size, and the files and file groups used to store it.

Notes

Before creating a database, consider that:

- Permission to create a database defaults to members of the **sysadmin** and **dbcreator** fixed server roles, although permissions can be granted to other users.
- The user who creates the database becomes the owner of the database.
- A maximum of 32,767 databases can be created on a server.
- The name of the database must follow the rules for identifiers.

Three types of files are used to store a database:

- ***Primary files:*** These files contain the startup information for the database. The primary files are also used to store data. Every database has one primary file.
- ***Secondary files:*** These files hold all the data that does not fit in the primary data file. Databases do not need secondary data files if the primary file is large enough to hold all the data in the database. Some databases may be large enough to need multiple secondary data files, or they may use secondary files on separate disk drives to spread the data across multiple disks.
- ***Transaction log:*** These files hold the log information used to recover the database. There must be at least one transaction log file for each database, although there may be more than one. The minimum size for a log file is 512 kilobytes (KB).

Caution Microsoft® SQL Server™ 2000 data and transaction log files must not be placed on compressed file systems or a remote network drive, such as a shared network directory.

When a database is created, all the files that comprise the database are filled with zeros to overwrite any existing data left on the disk by previously deleted files. Although this means that the files take longer to create, this action prevents the operating system from having to fill the files with zeros when data is written to the files for the first time during usual database operations. This improves the performance of day-to-day operations.

It is recommended that you specify a maximum size to which the file is permitted to grow. This prevents the file from growing, as data is added, until disk space is exhausted. To specify a maximum size for the file, use the MAXSIZE parameter of the CREATE DATABASE statement or the **Restrict filegrowth (MB)** option when using the **Properties** dialog box in SQL Server Enterprise Manager to create the database.

After you create a database, it is recommended that you create a backup of the **master** database.

Creating a Database

To create a database in SQL server you use either of the following tools:

1. Transact-SQL
2. Enterprise Manager
3. SQL-DMO

To create a database using the Create Database Wizard, you have to use the Enterprise Manager

Notes

Self Assessment

Fill in the blanks:

1. After you create a database, it is recommended that you create a ……. of the **master** database.
2. It is recommended that you specify a ……. to which the file is permitted to grow.
3. ……. columns provide a mechanism to identify each item stored in the table.
4. The user who creates the database becomes the ……. of the database.
5. ……. are files hold the log information used to recover the database. To create a database using the Create Database Wizard, you have to use the Enterprise Manager

5.5 Tables

Tables are database objects that contain all the data in a database. A table definition is a collection of columns. In tables, data is organized in a row-and-column format similar to a spreadsheet. Each row represents a unique record, and each column represents a field within the record. For example, a table containing employee data for a company can contain a row for each employee and columns representing employee information such as employee number, name, address, job title, and home phone number.

When you design a database, you decide what tables you need, what type of data goes in each table, who can access each table, and so on. As you create and work with tables, you continue to make more detailed decisions about them.

5.5.1 Designing Tables

The most efficient way to create a table is to define everything you need in the table at one time, including its data restrictions and additional components. However, you can also create a basic table, add some data to it, and then work with it for a while. This approach gives you a chance to see what types of transactions are most common and what types of data are frequently entered before you commit to a firm design by adding constraints, indexes, defaults, rules, and other objects.

It is a good idea to outline your plans on paper before creating a table and its objects. Decisions that must be made include:

- Types of data the table will contain.
- Columns in the table and the data type (and length, if required) for each column.
- Which columns accept null values.
- Whether and where to use constraints or defaults and rules.
- Types of indexes needed, where required, and which columns are primary keys and which are foreign keys.

5.5.2 Creating and Modifying a Table

After you have designed the database, the tables that will store the data in the database can be created. The data is usually stored in permanent tables. Tables are stored in the database files until they are deleted and are available to any user who has the appropriate permissions.

Notes

Temporary Tables

You can also create temporary tables. Temporary tables are similar to permanent tables, except temporary tables are stored in **tempdb** and are deleted automatically when no longer in use.

The two types of temporary tables, local and global, differ from each other in their names, their visibility, and their availability. Local temporary tables have a single number sign (#) as the first character of their names; they are visible only to the current connection for the user; and they are deleted when the user disconnects from instances of Microsoft® SQL Server™ 2000. Global temporary tables have two number signs (##) as the first characters of their names; they are visible to any user after they are created; and they are deleted when all users referencing the table disconnect from SQL Server.

For example, if you create a table named **employees**, the table can be used by any person who has the security permissions in the database to use it, until the table is deleted. If you create a local temporary table named **#employees**, you are the only person who can work with the table, and it is deleted when you disconnect. If you create a global temporary table named **##employees**, any user in the database can work with this table. If no other user works with this table after you create it, the table is deleted when you disconnect. If another user works with the table after you create it, SQL Server deletes it when both of you disconnect.

Table Properties

You can define up to 1,024 columns per table. Table and column names must follow the rules for identifiers; they must be unique within a given table, but you can use the same column name in different tables in the same database. You must also define a data type for each column.

Although table names must be unique for each owner within a database, you can create multiple tables with the same name if you specify different owners for each. You can create two tables named **employees** and designate **Jonah** as the owner of one and **Sally** as the owner of the other. When you need to work with one of the **employees** tables, you can distinguish between the two tables by specifying the owner with the name of the table.

Creating a Table

To create a table use can use any of the three tools in SQL server:

1. Transact-SQL
2. Enterprise Manager
3. SQL-DMO

Modifying Tables

After a table is created, you can change many of the options that were defined for the table when it was originally created, including:

- Columns can be added, modified, or deleted. For example, the column name, length, data type, precision, scale, and nullability can all be changed, although some restrictions exist.
- PRIMARY KEY and FOREIGN KEY constraints can be added or deleted.
- UNIQUE and CHECK constraints and DEFAULT definitions (and objects) can be added or deleted.

Notes

- An identifier column can be added or deleted using the IDENTITY or ROWGUIDCOL property. The ROWGUIDCOL property can also be added to or removed from an existing column, although only one column in a table can have the ROWGUIDCOL property at any one time.
- A table and selected columns within the table can be registered for full-text indexing.

5.6 Constraints

Constraints let you define the way the Database Engine automatically enforces the integrity of a database. Constraints define rules regarding the values allowed in columns and are the standard mechanism for enforcing integrity. Using constraints is preferred to using DML Triggers, rules, and defaults. The query optimizer also uses constraint definitions to build high-performance query execution plans.

5.6.1 Classes of Constraints

SQL Server supports the following classes of constraints:

- ***NOT NULL*** specifies that the column does not accept NULL values.
- ***CHECK*** constraints enforce domain integrity by limiting the values that can be put in a column.

  A CHECK constraint specifies a Boolean (evaluates to TRUE, FALSE, or unknown) search condition that is applied to all values that are entered for the column. All values that evaluate to FALSE are rejected. You can specify multiple CHECK constraints for each column. The following sample shows creating the constraint chk_id. This constraint additionally enforces the domain of the primary key by making sure that only numbers within a specified range are entered for the key.

  CREATE TABLE cust_sample

```
(
 cust_id            int      PRIMARY KEY,
 cust_name         char(50),
 cust_address         char(50),
 cust_credit_limit   money,
 CONSTRAINT chk_id CHECK (cust_id BETWEEN 0 and 10000 )
 )
```

- ***UNIQUE*** constraints enforce the uniqueness of the values in a set of columns.

  In a UNIQUE constraint, no two rows in the table can have the same value for the columns. Primary keys also enforce uniqueness, but primary keys do not allow for NULL as one of the unique values.
- ***PRIMARY KEY*** constraints identify the column or set of columns that have values that uniquely identify a row in a table.

Did u know? No two rows in a table can have the same primary key value. You cannot enter NULL for any column in a primary key. We recommend using a small, integer column as a primary key. Each table should have a primary key. A column or combination of columns that qualify as a primary key value is referred to as a candidate key.

Notes

The following example creates the part_sample table and specifies the part_nmbr field as the primary key.

```
CREATE TABLE part_sample
    (part_nmbr    int      PRIMARY KEY,
    part_name    char(30),
    part_weight    decimal(6,2),
    part_color    char(15) );
```

- FOREIGN KEY constraints identify and enforce the relationships between tables.

  A foreign key in one table points to a candidate key in another table. In the following example, the order_part table establishes a foreign key that references the part_sample table defined previously.

```
CREATE TABLE order_part
    (order_nmbr    int,
    part_nmbr    int
      FOREIGN KEY REFERENCES part_sample(part_nmbr)
       ON DELETE NO ACTION,
    qty_ordered    int);
GO
```

  You cannot insert a row with a foreign key value, except NULL, if there is no candidate key with that value. The ON DELETE clause controls what actions are taken when you try to delete a row to which existing foreign keys point. The ON DELETE clause has the following options:

  - NO ACTION specifies that the deletion fails with an error.
  - CASCADE specifies that all the rows with foreign keys pointing to the deleted row are also deleted.
  - SET NULL specifies that all rows with foreign keys pointing to the deleted row are set to NULL.
  - SET DEFAULT specifies that all rows with foreign keys pointing to the deleted row are set to their default value.

  The ON UPDATE clause defines the actions that are taken if you try to update a candidate key value to which existing foreign keys point. This clause also supports the NO ACTION, CASCADE, SET NULL and SET DEFAULT options.

5.6.2 Column and Table Constraints

Constraints can be column constraints or table constraints. A column constraint is specified as part of a column definition and applies only to that column. The constraints in the previous examples are column constraints. A table constraint is declared independently from a column definition and can apply to more than one column in a table. Table constraints must be used when more than one column must be included in a constraint.

For example, if a table has two or more columns in the primary key, you must use a table constraint to include both columns in the primary key. Consider a table that records events

Notes

occurring in a computer in a factory. Assume that events of several types can occur at the same time, but that no two events occurring at the same time can be of the same type. This can be enforced in the table by including both the event_type and event_time columns in a two-column primary key, as shown in the following example.

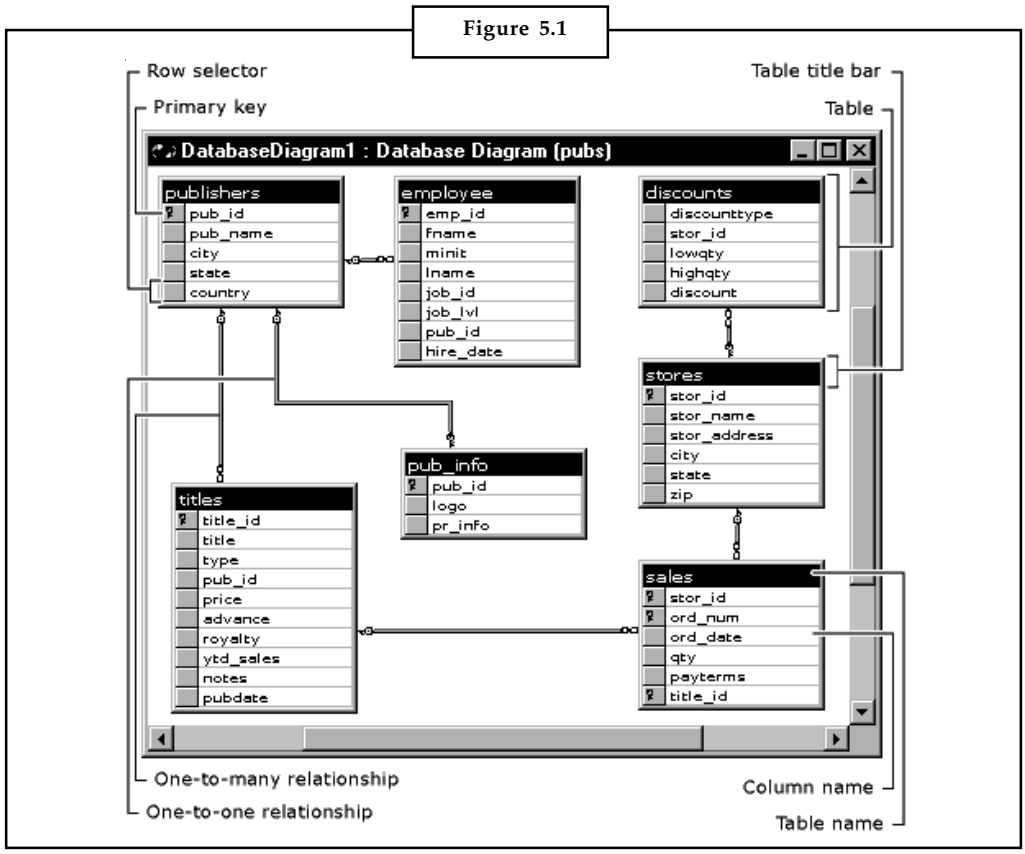
```
CREATE TABLE factory_process
  (event_type   int,
   event_time   datetime,
  event_site   char(50),
  event_desc   char(1024),
CONSTRAINT event_key PRIMARY KEY (event_type, event_time)
```

Task Differentiate between a column and table constraint with suitable examples.

5.7 Database Diagrams

The Database Designer is a visual tool that allows you to design and visualize a database to which you are connected. When designing a database, you can use Database Designer to create, edit, or delete tables, columns, keys, indexes, relationships, and constraints. To visualize a database, you can create one or more diagrams illustrating some or all of the tables, columns, keys, and relationships in it.

Figure 5.1



Notes

For any database, you can create as many database diagrams as you like; each database table can appear on any number of diagrams. Thus, you can create different diagrams to visualize different portions of the database, or to accentuate different aspects of the design. For example, you can create a large diagram showing all tables and columns, and you can create a smaller diagram showing all tables without showing the columns.

Each database diagram you create is stored in the associated database.

5.7.1 Creating Database Diagrams

You can use Object Explorer to create new database diagrams. Database diagrams graphically show the structure of the database. Using database diagrams you can create and modify tables, columns, relationships, and keys. Additionally, you can modify indexes and constraints.

To create a new database diagram

1. In Object Explorer, right-click the Database Diagrams folder or any diagram in that folder.
2. Choose New Database Diagram on the shortcut menu.

   The Add Table dialog box appears.
3. Select the required tables in the Tables list and click Add.

   The tables are displayed graphically in the new database diagram.

You can continue to add or delete tables, modify the existing tables, and alter table relationships until the new database diagram is complete.

5.8 Views

A view is a virtual table whose contents are defined by a query. Like a real table, a view consists of a set of named columns and rows of data. However, a view does not exist as a stored set of data values in a database. The rows and columns of data come from tables referenced in the query defining the view and are produced dynamically when the view is referenced.

A view acts as a filter on the underlying tables referenced in the view. The query that defines the view can be from one or more tables or from other views in the current or other databases. Distributed queries can also be used to define views that use data from multiple heterogeneous sources. This is useful, for example, if you want to combine similarly structured data from different servers each of which stores data for a different region of your organization.

Notes There are no restrictions on querying through views and few restrictions on modifying data through them.

This illustration shows a view based on two tables on next page.

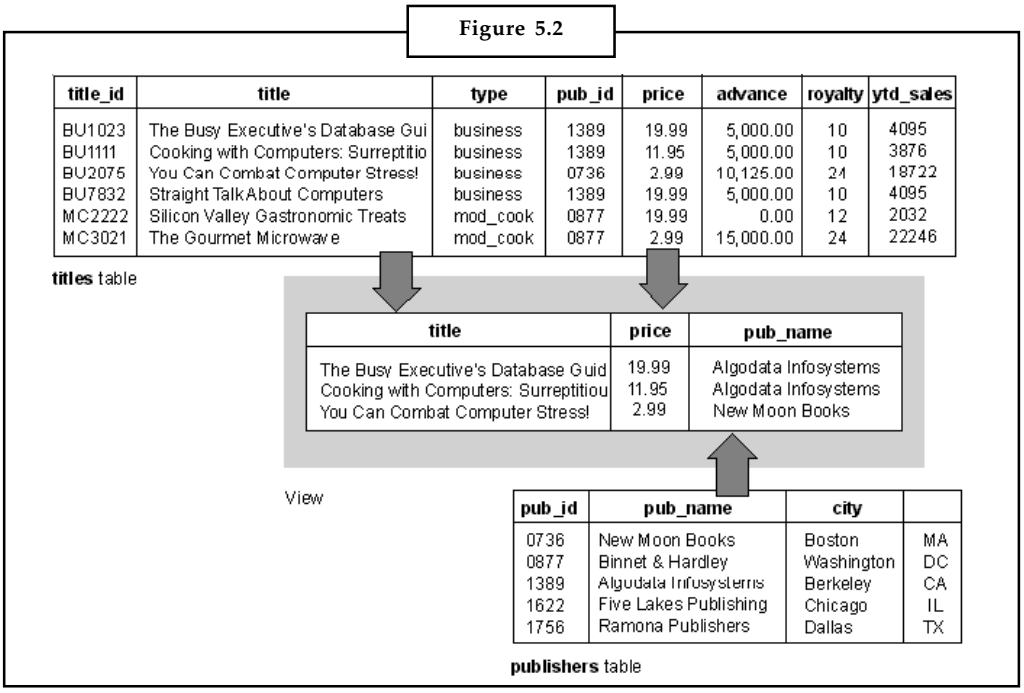
Before you create a view, consider these guidelines:

- You can create views only in the current database. However, the tables and views referenced by the new view can exist in other databases or even other servers if the view is defined using distributed queries.
- View names must follow the rules for identifiers and must be unique for each user. Additionally, the name must not be the same as any tables owned by that user.
- You can build views on other views and on procedures that reference views. Microsoft® SQL Server™ 2000 allows views to be nested up to 32 levels.

Notes

- You cannot associate rules or DEFAULT definitions with views.
- You cannot associate AFTER triggers with views, only INSTEAD OF triggers.
- The query defining the view cannot include the ORDER BY, COMPUTE, or COMPUTE BY clauses or the INTO keyword.
- You cannot define full-text index definitions on views.
- You cannot create temporary views, and you cannot create views on temporary tables.
- Views or tables participating in a view created with the SCHEMABINDING clause cannot be dropped, unless the view is dropped or changed so that it no longer has schema binding. In addition, ALTER TABLE statements on tables that participate in views having schema binding will fail if these statements affect the view definition.
- You cannot issue full-text queries against a view, although a view definition can include a full-text query if the query references a table that has been configured for full-text indexing.
- You must specify the name of every column in the view if:
- Any of the columns in the view is derived from an arithmetic expression, a built-in function, or a constant.
- Two or more of the columns in the view would otherwise have the same name (usually because the view definition includes a join and the columns from two or more different tables have the same name).
- You want to give any column in the view a name different from the column from which it is derived. (You can also rename columns in the view.) A view column inherits the data type of the column from which it is derived, whether or not you rename it.

Figure 5.2



title_id	title	type	pub_id	price	advance	royalty	ytd_sales
BU1023	The Busy Executive's Database Gui	business	1389	19.99	5,000.00	10	4095
BU1111	Cooking with Computers: Surreptitio	business	1389	11.95	5,000.00	10	3876
BU2075	You Can Combat Computer Stress!	business	0736	2.99	10,125.00	24	18722
BU7832	Straight Talk About Computers	business	1389	19.99	5,000.00	10	4095
MC2222	Silicon Valley Gastronomic Treats	mod_cook	0877	19.99	0.00	12	2032
MC3021	The Gourmet Microwave	mod_cook	0877	2.99	15,000.00	24	22246

title	price	pub_name
The Busy Executive's Database Guid	19.99	Algodata Infosystems
Cooking with Computers: Surreptitiou	11.95	Algodata Infosystems
You Can Combat Computer Stress!	2.99	New Moon Books

pub_id	pub_name	city	
0736	New Moon Books	Boston	MA
0877	Binnet & Hardley	Washington	DC
1389	Algodata Infosystems	Berkeley	CA
1622	Five Lakes Publishing	Chicago	IL
1756	Ramona Publishers	Dallas	TX

Notes This rule does not apply when a view is based on a query containing an outer join because columns may change from not allowing null values to allowing them.

Notes

Otherwise, you do not need to specify column names when creating the view. SQL Server gives the columns of the view the same names and data types as the columns to which the query defining the view refers. The select list can be a full or partial list of the column names in the base tables.

Notes To create a view you must be granted permission to do so by the database owner and you must have appropriate permissions on any tables or views referenced in the view definition.

By default, as rows are added or updated through a view, they disappear from the scope of the view when they no longer fall into the criteria of the query defining the view. For example, a query can be created, defining a view that retrieves all rows from a table where the employee's salary is less than ₹ 30,000. If the employee's salary is increased to ₹ 32,000, then querying the view no longer displays that particular employee because his or her salary does not conform to the criteria set by the view. However, the WITH CHECK OPTION clause forces all data modification statements executed against the view to adhere to the criteria set within the SELECT statement defining the view. If you use this clause, rows cannot be modified in a way that causes them to disappear from the view. Any modification that would cause this to happen is canceled and an error is displayed.

The definition of a sensitive view can be obfuscated by using the WITH ENCRYPTION option. Note that obfuscated view definitions can be reverse engineered because SQL Server must de-obfuscate them when they are executed. In SQL Server 2000, the obfuscated text is visible in the **syscomments** system table and may be susceptible to de-obfuscation attempts.

Creating a View

To create a view, you can use any of the tools given below:

1. Transact-SQL
2. Enterprise Manager
3. SQL-DMO

You can also create a view using the SQL Server Enterprise Manager Create View Wizard.

Self Assessment

Name the following:

6. A virtual table whose contents are defined by a query.
7. A visual tool that allows you to design and visualize a database to which you are connected.
8. A type of constraint that is declared independently from a column definition and can apply to more than one column in a table.
9. A key in one table that points to a candidate key in another table.
10. Keys that enforce uniqueness.

5.9 Synonyms

Microsoft SQL Server introduces the concept of a synonym. A synonym is an alternative name for a schema-scoped object. Client applications can use a single-part name to reference a base object by using a synonym instead of using a two-part, three-part, or four-part name to reference the base object.

Notes

A synonym is a database object that serves the following purposes:

- Provides an alternative name for another database object, referred to as the base object, that can exist on a local or remote server.
- Provides a layer of abstraction that protects a client application from changes made to the name or location of the base object.

For example, consider the Employee table of the AdventureWorks2008R2 sample database, located on a server named Server1. To reference this table from another server, Server2, a client application would have to use the four-part name Server1. Adventure Works. Human Resources. Employee. Also, if the location of the table were to change, for example, to another server, the client application would have to be modified to reflect that change.

To address both these issues, you can create a synonym, EmpTable, on Server2 for the Employee table on Server1. Now, the client application only has to use the single-part name, EmpTable, to reference the Employee table. Also, if the location of the Employee table changes, you will have to modify the synonym, EmpTable, to point to the new location of the Employee table. Because there is no ALTER SYNONYM statement, you first have to drop the synonym, EmpTable, and then re-create the synonym with the same name, but point the synonym to the new location of Employee.

5.9.1 Syntax for Creating Synonym

CREATE SYNONYM [schema_name_1.] synonym_name FOR <object>

<object> :: =

{

[server_name.[database_name] . [schema_name_2].| database_name . [schema_name_2].| schema_name_2.] object_name

}

5.9.2 Arguments

schema_name_1

Specifies the schema in which the synonym is created. If schema is not specified, SQL Server uses the default schema of the current user.

synonym_name

Is the name of the new synonym.

server_name

Is the name of the server on which base object is located.

database_name

Is the name of the database in which the base object is located. If database_name is not specified, the name of the current database is used.

schema_name_2

Is the name of the schema of the base object. If schema_name is not specified the default schema of the current user is used.

object_name

Is the name of the base object that the synonym references.

Notes

5.9.3 Permissions

To create a synonym in a given schema, a user must have CREATE SYNONYM permission and either own the schema or have ALTER SCHEMA permission.

The CREATE SYNONYM permission is a grantable permission.

Notes You do not need permission on the base object to successfully compile the CREATE SYNONYM statement, because all permission checking on the base object is deferred until run time.

5.10 Programming Objects

There are a number of programming objects such as functions, triggers, procedures etc., used in SQL server. In this topic, you learn how to create these programming objects in SQL Server.

5.10.1 Creating Procedures

When you create an application with Microsoft® SQL Server™ 2000, the Transact-SQL programming language is the primary programming interface between your applications and the SQL Server database. When you use Transact-SQL programs, two methods are available for storing and executing the programs. You can store the programs locally and create applications that send the commands to SQL Server and process the results, or you can store the programs as stored procedures in SQL Server and create applications that execute the stored procedures and process the results.

Stored procedures in SQL Server are similar to procedures in other programming languages in that they can:

- Accept input parameters and return multiple values in the form of output parameters to the calling procedure or batch.
- Contain programming statements that perform operations in the database, including calling other procedures.
- Return a status value to a calling procedure or batch to indicate success or failure (and the reason for failure).

You can use the Transact-SQL EXECUTE statement to run a stored procedure. Stored procedures are different from functions in that they do not return values in place of their names and they cannot be used directly in an expression.

The benefits of using stored procedures in SQL Server rather than Transact-SQL programs stored locally on client computers are:

- ***They allow modular programming:*** You can create the procedure once, store it in the database, and call it any number of times in your program. Stored procedures can be created by a person who specializes in database programming, and they can be modified independently of the program source code.
- ***They allow faster execution:*** If the operation requires a large amount of Transact-SQL code or is performed repetitively, stored procedures can be faster than batches of Transact-SQL code. They are parsed and optimized when they are first executed, and a compiled version of the stored procedure remains in memory cache for later use. This means the stored

Notes

procedure does not need to be reparsed and reoptimized with each use resulting in much faster execution times.

- ***They can reduce network traffic:*** An operation requiring hundreds of lines of Transact-SQL code can be performed through a single statement that executes the code in a procedure, rather than by sending hundreds of lines of code over the network.
- ***They can be used as a security mechanism:*** Users can be granted permission to execute a stored procedure even if they do not have permission to execute the procedure's statements directly.

A SQL Server stored procedure is created with the Transact-SQL CREATE PROCEDURE statement and can be modified with the ALTER PROCEDURE statement. The stored procedure definition contains two primary components: the specification of the procedure name and its parameters, and the body of the procedure, which contains Transact-SQL statements that perform the procedure's operations.

Prerequisites for Creating Stored Procedure

You can create stored procedures using the CREATE PROCEDURE Transact-SQL statement. Before creating a stored procedure, consider that:

- CREATE PROCEDURE statements cannot be combined with other SQL statements in a single batch.
- Permission to create stored procedures defaults to the database owner, who can transfer it to other users.
- Stored procedures are database objects, and their names must follow the rules for identifiers.
- You can create a stored procedure only in the current database.

When creating a stored procedure, you should specify:

- Any input parameters and output parameters to the calling procedure or batch.
- The programming statements that perform operations in the database, including calling other procedures.
- The status value returned to the calling procedure or batch to indicate success or failure (and the reason for failure).

Example: Let us explain how to create and use Stored Procedures with Input Parameters and output parameters as below.

Creating Stored Procedure with Input Parameters

Input Parameters in Stored Procedures are considered as placeholders for data that the user requires to send. Technically, input parameters are memory variables since they are accumulated in memory.

For creating Stored Procedure with Input Parameters, code is written as below.

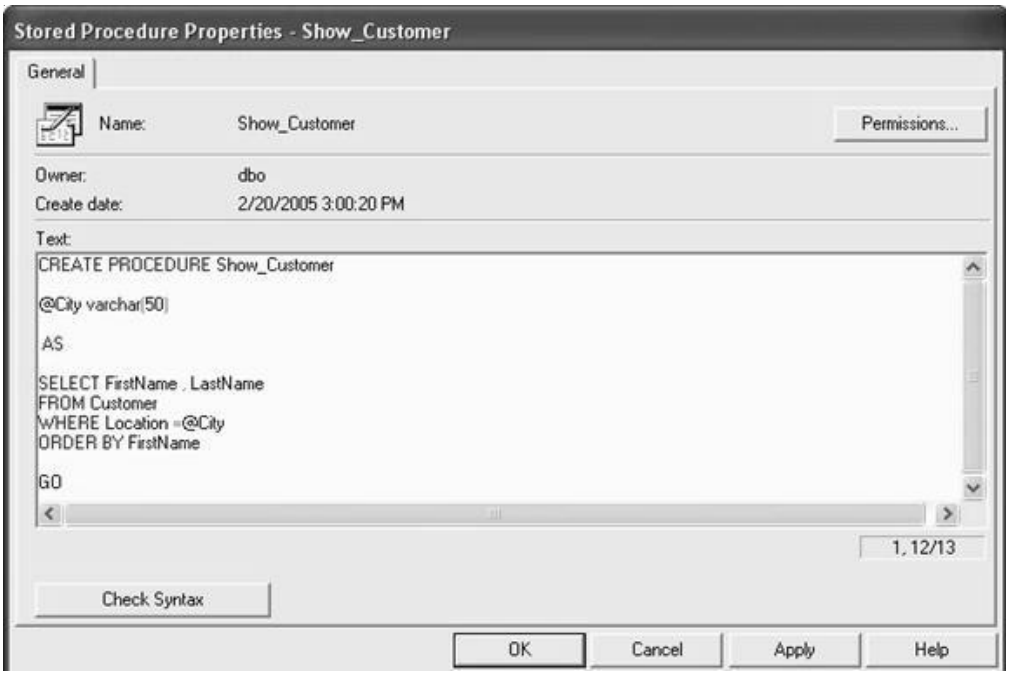
```
CREATE PROCEDURE Show_Customer

@City varchar(50)

AS

SELECT FirstName, LastName FROM Customer
```

Notes

WHERE Location=@City

ORDER BY FirstName

Here the placeholder i.e. Input parameter is @City variable. It acknowledges the value, sends by the caller program and implements the procedure according to it. Here, we can obtain all the Customer's First and Last Names having location values located in @City variable.

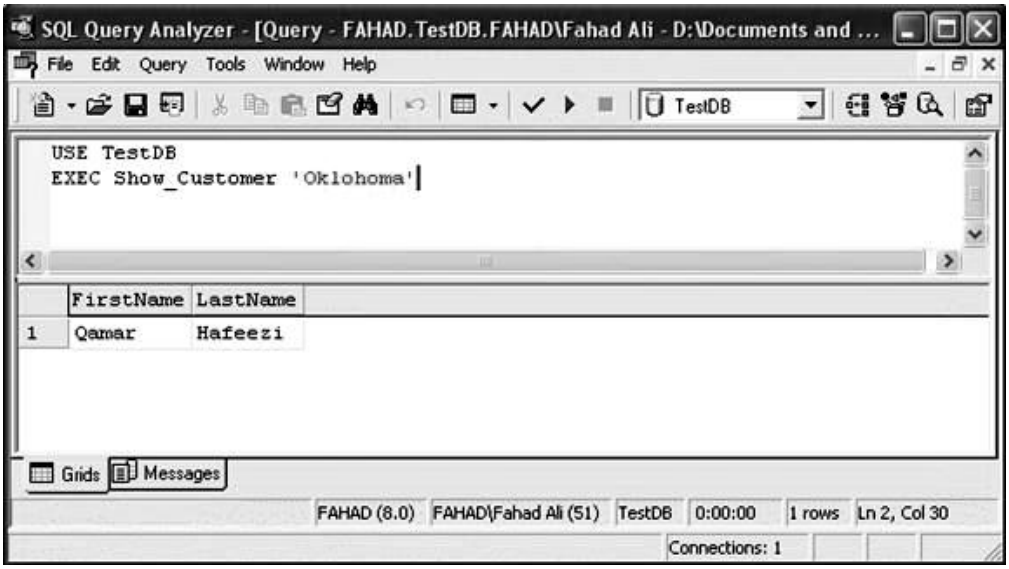


For executing, use the code given below in Query Analyzer

USE TestDB

EXEC Show_Customer 'Oklahoma'

Here we have modified the city from "New York" to "Oklahoma" and now our stored procedure is now well generalized to return the Customers for variable cities, based on the input value.



Notes

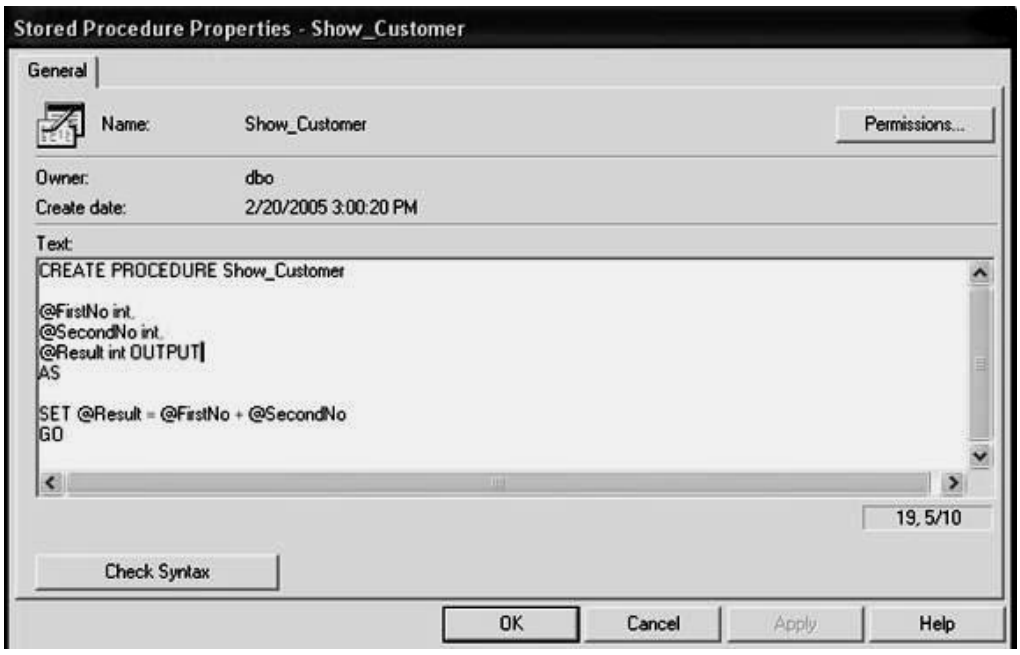
Creating Stored Procedure with Output Parameters

Now we will use Output parameters in our Stored Procedure to return some value from it. Output parameters are created in the equivalent space as the input parameters, right among the procedure name and AS sections. The only dissimilarity is that they are specified with the word OUTPUT immediately afterward.

Open window for New Stored Procedure and add the below code in it,

```
CREATE PROCEDURE Show_Customer
@FirstNo varchar(50),
@SecondNo varchar(50),
@Result varchar(50) OUTPUT
AS
SET @Result = @FirstNo + @SecondNo
```

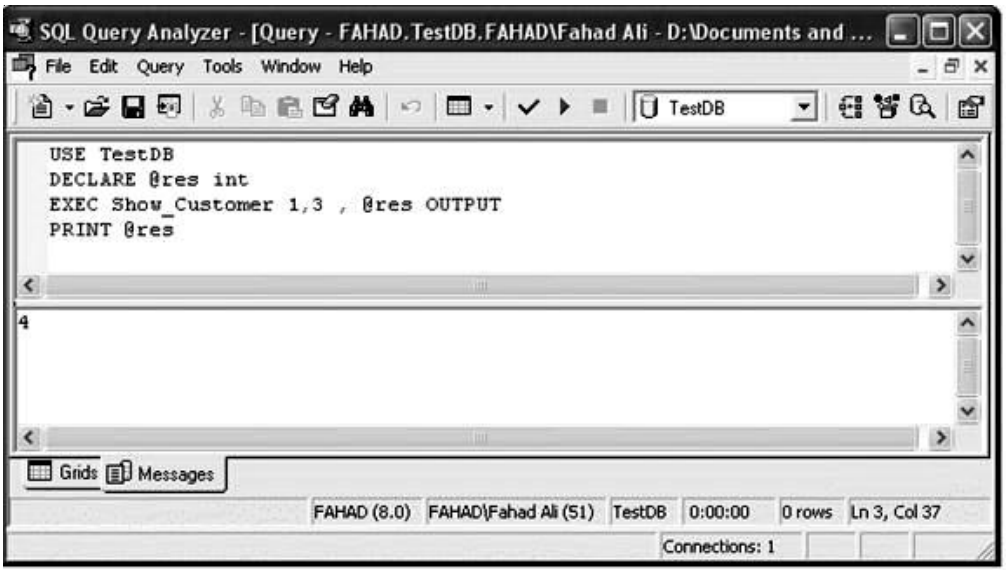
Let us explain another type of example because of the reason that it is much simple to understand how OUTPUT parameters work.



Implementing the Stored Procedure having OUTPUT parameters is very simple. Create a local variable to hold the returned value and exhibit it. Here is the code

```
USE TestDB
DECLARE @res int
EXEC Show_Customer 1,3 , @res OUTPUT
PRINT @res
```

Notes



5.10.2 Functions

Functions are subroutines made up of one or more Transact-SQL statements that can be used to encapsulate code for reuse. Microsoft® SQL Server™ 2000 does not limit users to the built-in functions defined as part of the Transact-SQL language, but allows users to create their own user-defined functions.

User-defined functions are created using the CREATE FUNCTION statement, modified using the ALTER FUNCTION statement, and removed using the DROP FUNCTION statement. Each fully qualified user-defined function name (*database_name.owner_name.function_name*) must be unique.

You must have been granted CREATE FUNCTION permissions to create, alter, or drop user-defined functions. Users other than the owner must be granted appropriate permissions on a function before they can use it in a Transact-SQL statement. To create or alter tables with references to user-defined functions in the CHECK constraint, DEFAULT clause, or computed column definition, you must also have REFERENCES permission on the functions.

Transact-SQL errors that cause a statement to be canceled and continue with the next statement in the module (such as triggers or stored procedures) are treated differently inside a function. In functions, such errors cause the execution of the function to stop. This in turn causes the statement that invoked the function to be canceled.

Types of User – Defined Functions

SQL Server 2000 supports three types of user-defined functions:

- Scalar functions
- Inline table-valued functions
- Multistatement table-valued functions

Notes

Example: Creating and Calling T-SQL User-Defined Functions

User-Defined Functions, or UDFs, are considered as database objects that directly imitate the semantics of functions in programming languages. Similar to a function in C#, UDFs can comprise a variable number of input parameters and return a value of a specific type. A UDF can return either scalar data – a string, an integer, and so forth - or tabular data.

The following UDF computes the predictable value of the inventory for a specific product. It does so by taking in three input parameters – the UnitPrice, UnitsInStock, and Discontinued values for a specific product – and returns a value of type money. It calculates the expected value of the inventory by multiplying the UnitPrice by the UnitsInStock. For terminate items, this value is halved.

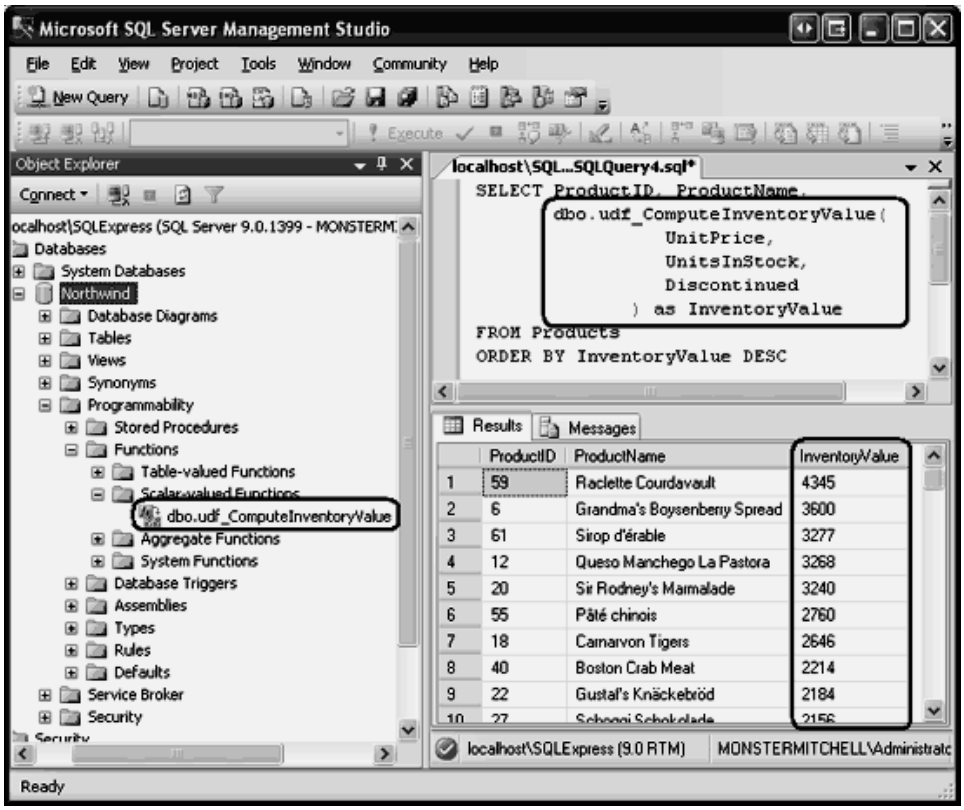
```
CREATE FUNCTION udf_ComputeInventoryValue
(
@UnitPrice money,
@UnitsInStock smallint,
@Discontinued bit
)
RETURNS money
AS
BEGIN
DECLARE @Value decimal
SET @Value = ISNULL(@UnitPrice, 0) * ISNULL(@UnitsInStock, 0)
IF @Discontinued = 1
SET @Value = @Value * 0.5
RETURN @Value
END
```

Once this UDF has been added to the database, it can be located via Management Studio by increasing the Programmability folder, then Functions, and then Scalar-value Functions. It can be accessed in a SELECT query like so:

```
SELECT ProductID, ProductName, dbo.udf_ComputeInventoryValue
(UnitPrice, UnitsInStock, Discontinued) as InventoryValue
FROM Products
ORDER BY InventoryValue DESC
```

We have added the udf_ComputeInventoryValue UDF to the Northwind database; Following Figure displays the output of the above SELECT query when observed via Management Studio. Make sure that the UDF is listed under the Scalar-value Functions folder in the Object Explorer.

Notes



UDFs can also return tabular data. For instance, we can create a UDF that returns products that relates to a specific category:

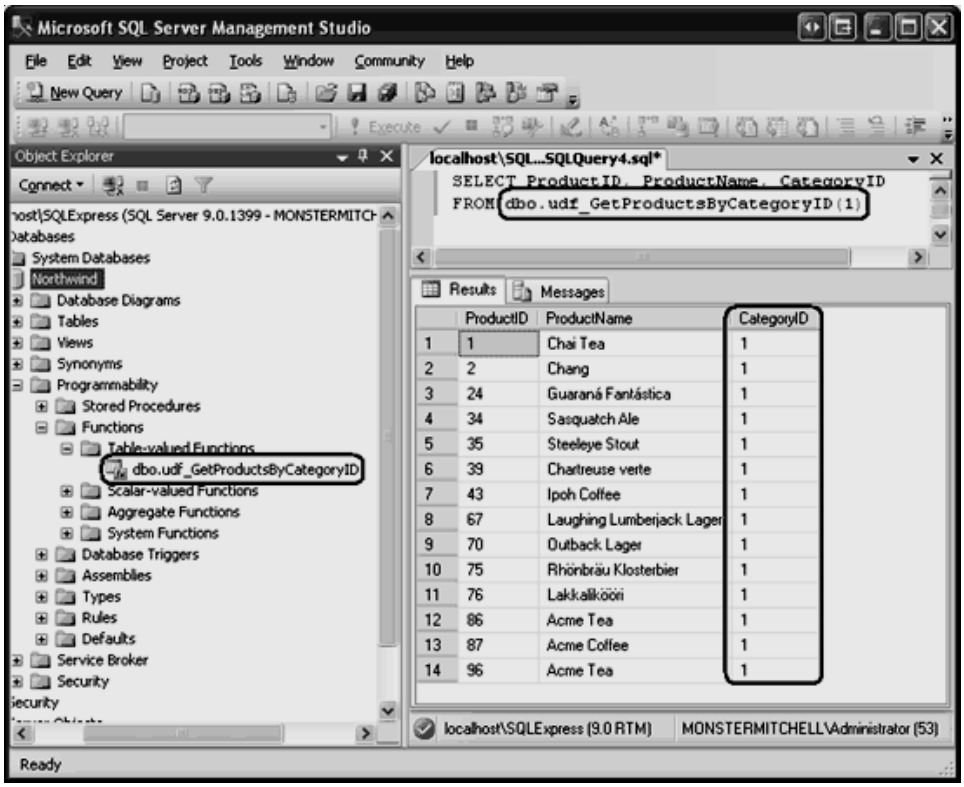
```
CREATE FUNCTION dbo.udf_GetProductsByCategoryID
(
    @CategoryID int
)
RETURNS TABLE
AS
RETURN
(
SELECT ProductID, ProductName, SupplierID, CategoryID, QuantityPerUnit, UnitPrice, UnitsInStock, UnitsOnOrder, ReorderLevel, Discontinued FROM Products WHERE CategoryID = @CategoryID
)
```

The udf_GetProductsByCategoryID UDF accepts a @CategoryID input parameter and returns the results of the particular SELECT query. Once created, this UDF can be referenced in the FROM (or JOIN) clause of a SELECT query. The example given below would return the ProductID, ProductName, and CategoryID values for all of the beverages.

```
SELECT ProductID, ProductName, CategoryID
FROM dbo.udf_GetProductsByCategoryID(1)
```

Notes

We have added the udf_GetProductsByCategoryID UDF to the Northwind database; Figure below displays the output of the above SELECT query when observed via Management Studio. UDFs that return tabular data can be located in the Object Explorer s Table-value Functions folder.



5.10.3 Triggers

Microsoft® SQL Server™ 2000 provides two primary mechanisms for enforcing business rules and data integrity: constraints and triggers. A trigger is a special type of stored procedure that automatically takes effect when the data in a specified table is modified. A trigger is invoked in response to an INSERT, UPDATE, or DELETE statement. A trigger can query other tables and can include complex Transact-SQL statements. The trigger and the statement that fires it are treated as a single transaction, which can be rolled back from within the trigger. If a severe error is detected (for example, insufficient disk space), the entire transaction automatically rolls back.

Triggers are useful in these ways:

- Triggers can cascade changes through related tables in the database; however, these changes can be executed more efficiently using cascading referential integrity constraints.
- Triggers can enforce restrictions that are more complex than those defined with CHECK constraints.

  Unlike CHECK constraints, triggers can reference columns in other tables. For example, a trigger can use a SELECT from another table to compare to the inserted or updated data and to perform additional actions, such as modify the data or display a user-defined error message.
- Triggers can also evaluate the state of a table before and after a data modification and take action(s) based on that difference.

Notes

- Multiple triggers of the same type (INSERT, UPDATE, or DELETE) on a table allow multiple, different actions to take place in response to the same modification statement.

Triggers Compared to Constraints

Constraints and triggers each have benefits that make them useful in special situations. The primary benefit of triggers is that they can contain complex processing logic that uses Transact-SQL code. Therefore, triggers can support all of the functionality of constraints; however, triggers are not always the best method for a given feature.

Entity integrity should always be enforced at the lowest level by indexes that are part of PRIMARY KEY and UNIQUE constraints or are created independently of constraints. Domain integrity should be enforced through CHECK constraints, and Referential Integrity (RI) should be enforced through FOREIGN KEY constraints, assuming their features meet the functional needs of the application.

Triggers are most useful when the features supported by constraints cannot meet the functional needs of the application. For example:

- FOREIGN KEY constraints can validate a column value only with an exact match to a value in another column, unless the REFERENCES clause defines a cascading referential action.
- A CHECK constraint can validate a column value only against a logical expression or another column in the same table. If your application requires that a column value be validated against a column in another table, you must use a trigger.
- Constraints can communicate about errors only through standardized system error messages. If your application requires (or can benefit from) customized messages and more complex error handling, you must use a trigger.

Triggers can cascade changes through related tables in the database; however, these changes can be executed more efficiently through cascading referential integrity constraints.

- Triggers can disallow or roll back changes that violate referential integrity, thereby canceling the attempted data modification. Such a trigger might go into effect when you change a foreign key and the new value does not match its primary key. For example, you can create an insert trigger on **titleauthor.title_id** that rolls back an insert if the new value does not match some value in **titles.title_id**. However, FOREIGN KEY constraints are usually used for this purpose.
- If constraints exist on the trigger table, they are checked after the INSTEAD OF trigger execution but prior to the AFTER trigger execution. If the constraints are violated, the INSTEAD OF trigger actions are rolled back and the AFTER trigger is not executed

Creating Trigger

Before you create a trigger, consider that:

- The CREATE TRIGGER statement must be the first statement in the batch. All other statements that follow in that batch are interpreted as part of the definition of the CREATE TRIGGER statement.
- Permission to create triggers defaults to the table owner, who cannot transfer it to other users.
- Triggers are database objects, and their names must follow the rules for identifiers.
- You can create a trigger only in the current database, although a trigger can reference objects outside of the current database.

Notes

- A trigger cannot be created on a temporary or system table, although triggers can reference temporary tables. System tables should not be referenced; use the Information Schema Views instead.
- INSTEAD OF DELETE and INSTEAD OF UPDATE triggers cannot be defined on a table that has a foreign key defined with a DELETE or UPDATE action.
- Although a TRUNCATE TABLE statement is like a DELETE statement without a WHERE clause (it deletes all rows), it does not cause DELETE triggers to fire because the TRUNCATE TABLE statement is not logged.
- The WRITETEXT statement does not cause the INSERT or UPDATE triggers to fire.

When you create a trigger, specify:

- The name.
- The table upon which the trigger is defined.
- When the trigger is to fire.
- The data modification statements that activate the trigger. Valid options are INSERT, UPDATE, or DELETE. More than one data modification statement can activate the same trigger. For example, a trigger can be activated by an INSERT and an UPDATE statement.
- The programming statements that perform the trigger action.

Multiple Triggers

A table can have multiple AFTER triggers of a given type provided they have different names; each trigger can perform numerous functions. However, each trigger can apply to only one table, although a single trigger can apply to any subset of three user actions (UPDATE, INSERT, and DELETE).

A table can have only one INSTEAD OF trigger of a given type.

Trigger Permissions and Ownership

CREATE TRIGGER permissions default to the table owner on which the trigger is defined, the **sysadmin** fixed server role, and members of the **db_owner** and **db_ddladmin** fixed database roles, and are not transferable.

If an INSTEAD OF trigger is created on a view, the ownership chain is broken if the view owner does not also own the base tables referenced by the view and trigger. For a base table not owned by the view owner, the table owner must separately grant the necessary permissions to anybody reading or updating the view. If the same user owns both the view and the underlying base tables, they have to grant other users permissions only on the view, not individual base tables.

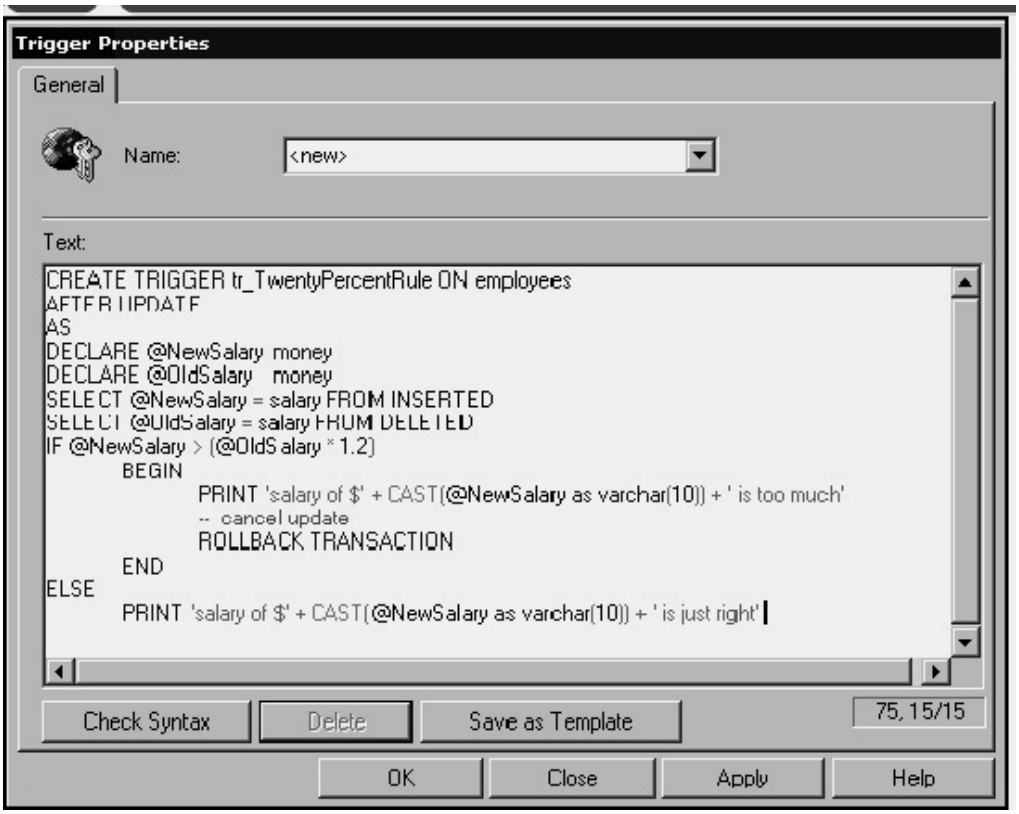
Creating Triggers

To create a trigger, you can use any of the following tools given below:

1. Transact-SQL
2. Enterprise Manager
3. SQL-DMO

Notes

Example: Let us consider that in your database there is a table into which you record data regarding your employees salaries, and that you are nor required to provide your employees increments of more than twenty percent at a time. You can handle this action with a trigger. Figure below show you how to create a trigger by means of the Enterprise Manager console (you can run the same code from the Query Analyzer):



If you are updating the record from the Query Analyzer window the suitable message will be printed; otherwise, you can select to raise an error to a calling application or just mutely log this error into a table. This Transact-SQL code uses the thought of transaction. It is important to recognize now that the operation (INSERT in my sample) and the trigger include a single block that either succeeds or fails as a whole; ROLL-BACK TRANSACTION rolls back the whole transaction, nullifying the update the data in the Employees table will remain as it was before the update. SQL Server also introduces two virtual tables, INSERTED and DELETED, for use with triggers. When data is about to be customized by insertion, updating, or deletion there is no assurance that the operation will be accomplished: Some business logic executed as constraints or triggers may avert this. SQL Server creates INSERTED and DELETED automatically whenever a particular action appears. The DELETED table comprises rows as they were prior to the alteration, and the INSERTED table includes modified rows, as does the base table itself. The number of rows in each table is comparable exactly the number of rows affected by the T-SQL statement.

5.11 Summary

- Designing a database requires an understanding of the business functions you want to model. It also requires an understanding of the database concepts and features that you want to use to represent those business functions.

Notes

- In planning the database, regardless of its size and complexity, use the following basic steps:
  - Gather information
  - Identify the objects
  - Model the objects
  - Identify the types of information for each object
  - Identify the relationships between objects.
- To create a database determine the name of the database, its owner (the user who creates the database), its size, and the files and file groups used to store it.
- Three types of files are used to store a database: Primary files, Secondary Files and Transaction Log.
- To create a database in SQL server you use either of the following tools:
  - Transact-SQL
  - Enterprise Manager
  - SQL-DMO
- Tables are database objects that contain all the data in a database. A table definition is a collection of columns. In tables, data is organized in a row-and-column format similar to a spreadsheet. Each row represents a unique record, and each column represents a field within the record.
- Constraints define rules regarding the values allowed in columns and are the standard mechanism for enforcing integrity.
- The Database Designer is a visual tool that allows you to design and visualize a database to which you are connected.
- A view is a virtual table whose contents are defined by a query. Like a real table, a view consists of a set of named columns and rows of data.
- When you use Transact-SQL programs, two methods are available for storing and executing the programs. You can store the programs locally and create applications that send the commands to SQL Server and process the results, or you can store the programs as stored procedures in SQL Server and create applications that execute the stored procedures and process the results.

5.12 Keywords

Categorical Columns: These columns classify or group the data and store a limited selection of data such as true/false, married/single, and VP/Director/Group Manager.

Check Constraint: A CHECK constraint specifies a Boolean (evaluates to TRUE, FALSE, or unknown) search condition that is applied to all values that are entered for the column.

Constraints: Constraints define rules regarding the values allowed in columns and are the standard mechanism for enforcing integrity.

Databases: A database in SQL Server is made up of a collection of tables.

FOREIGN KEY constraints: FOREIGN KEY constraints identify and enforce the relationships between tables.

Notes

Functions: Functions are subroutines made up of one or more Transact-SQL statements that can be used to encapsulate code for reuse.

Identifier Columns: These columns provide a mechanism to identify each item stored in the table.

Primary Key Constraint: PRIMARY KEY constraints identify the column or set of columns that have values that uniquely identify a row in a table.

Raw Data Columns: These columns store tangible pieces of information, such as names, determined by a source external to the database.

Tables: Tables are database objects that contain all the data in a database.

Triggers: A trigger is a special type of stored procedure that automatically takes effect when the data in a specified table is modified.

UNIQUE Constraints: A constraint that enforce the uniqueness of the values in a set of columns.

View: A view is a virtual table whose contents are defined by a query

5.13 Review Questions

1. Discuss the various types of system databases in SQL Server.
2. How can a user modify the system data in SQL server? Explain in detail.
3. Describe in detail the various steps involved in database planning.
4. Discuss the process of creating databases in SQL Server.
5. Comment: It is recommended that you specify a maximum size to which the file is permitted to grow.
6. What are tables? Describe the procedure of creating and modifying a table in SQL server.
7. Analyze the statement: Using constraints is preferred to using DML Triggers, rules, and defaults with suitable reasons.
8. Explain the benefits of using stored procedures in SQL Server rather than Transact-SQL programs stored locally on client computers.
9. How can you create database diagrams in SQL Server?
10. Enlist the important guidelines before you create a view.
11. Describe the prerequisites for Creating Stored Procedure.
12. Explain briefly the procedure of creating following objects in SQL server:
    (a) Functions
    (b) Procedures
    (c) Triggers

Answers: Self Assessment

1.	Backup	2.	Maximum size
3.	Identifier columns	4.	Owner
5.	Transaction log	6.	Views

Notes

7. Database Designers
8. Table constraint
9. Foreign Key
10. Unique Key

5.14 Further Readings

Books

C.J. Date, *Introduction to Database Systems*, Pearson Education.

Elmasri Navrate, *Fundamentals of Database Systems*, Pearson Education.

Peter Rob & Carlos Coronel, *Database Systems Design, Implementation and Management*, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, *Database Management Systems*, Tata McGraw Hill, 3rd Edition.

Silberschatz, Korth, *Database System Concepts*, McGraw Hill, 5th Edition.

Online links

www.sql-tutorial.net/

www.w3schools.com/sql/

www.sqlservertutorials.com/

Notes

Unit 6: SQL Server Authentication

CONTENTS

Objectives
Introduction
6.1 Authentication Modes
6.1.1 Windows Authentication
6.1.2 SQL Server Authentication
6.2 Permissions
6.2.1 Permissions Applicable to Specific Securables
6.2.2 Permissions Validation
6.3 Disadvantages of SQL Server Authentication
6.4 Advantages of SQL Server Authentication
6.5 SQL Server Encryption
6.5.1 Encryption Hierarchy
6.5.2 Encryption Mechanisms
6.6 Summary
6.7 Keywords
6.8 Review Questions
6.9 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand the basic principles of authentication
- Learn about various permissions in SQL server authentication
- Analyze the process of SQL server encryption

Introduction

A user passes through two stages of security when working in Microsoft® SQL Server™: authentication and authorization (permissions validation). The authentication stage identifies the user using a login account and verifies only the ability to connect to an instance of SQL Server. If authentication is successful, the user connects to an instance of SQL Server. The user then needs permissions to access databases on the server, which is done by granting access to an account in each database, mapped to the user login. The permissions validation stage controls the activities the user is allowed to perform in the SQL Server database.

6.1 Authentication Modes

Microsoft SQL Server can operate in one of two security (authentication) modes:

1. ***Windows Authentication Mode (Windows Authentication):*** Windows Authentication mode allows a user to connect through a Microsoft Windows NT® 4.0 or Windows® 2000 user account.

Notes

2. ***Mixed Mode (Windows Authentication and SQL Server Authentication):*** Mixed Mode allows users to connect to an instance of SQL Server using either Windows Authentication or SQL Server Authentication. Users who connect through a Windows NT 4.0 or Windows 2000 user account can make use of trusted connections in either Windows Authentication Mode or Mixed Mode.

SQL Server Authentication is provided for backward compatibility. For example, if you create a single Windows 2000 group and add all necessary users to that group you will need to grant the Windows 2000 group login rights to SQL Server and access to any necessary databases.

Security Note: When possible, use Windows Authentication.

6.1.1 Windows Authentication

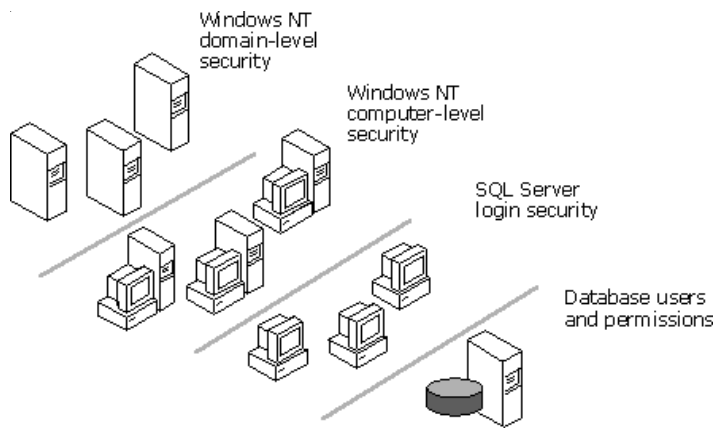
When a user connects through a Windows NT 4.0 or Windows 2000 user account, SQL Server revalidates the account name and password by calling back to Windows NT 4.0 or Windows 2000 for the information.

SQL Server achieves login security integration with Windows NT 4.0 or Windows 2000 by using the security attributes of a network user to control login access. A user's network security attributes are established at network login time and are validated by a Windows domain controller. When a network user tries to connect, SQL Server uses Windows-based facilities to determine the validated network user name. SQL Server then verifies that the person is who they say they are, and then permits or denies login access based on that network user name alone, without requiring a separate login name and password.

Login security integration operates over any supported network protocol in SQL Server.

Notes If a user attempts to connect to an instance of SQL Server providing a blank login name, SQL Server uses Windows Authentication. Additionally, if a user attempts to connect to an instance of SQL Server configured for Windows Authentication Mode by using a specific login, the login is ignored and Windows Authentication is used.

Figure 6.1



Windows Authentication has certain benefits over SQL Server Authentication, primarily due to its integration with the Windows NT 4.0 and Windows 2000 security system. Windows NT 4.0 and Windows 2000 security provides more features, such as secure validation and encryption of

Notes

passwords, auditing, password expiration, minimum password length, and account lockout after multiple invalid login requests.

Because Windows NT 4.0 and Windows 2000 users and groups are maintained only by Windows NT 4.0 or Windows 2000, SQL Server reads information about a user's membership in groups when the user connects. If changes are made to the accessibility rights of a connected user, the changes become effective the next time the user connects to an instance of SQL Server or logs on to Windows NT 4.0 or Windows 2000 (depending on the type of change).

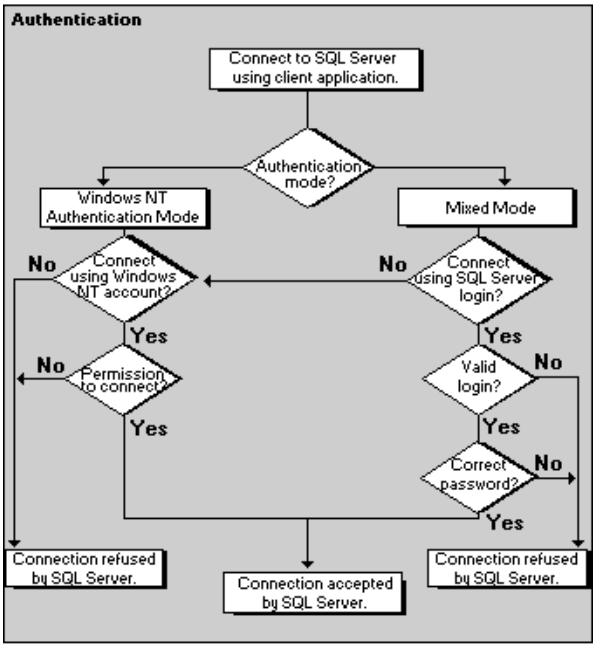
Notes Windows Authentication Mode is not available when an instance of SQL Server is running on Windows 98 or Microsoft Windows Millennium Edition.

6.1.2 SQL Server Authentication

When a user connects with a specified login name and password from a nontrusted connection, SQL Server performs the authentication itself by checking to see if a SQL Server login account has been set up and if the specified password matches the one previously recorded. If SQL Server does not have a login account set, authentication fails and the user receives an error message.

SQL Server Authentication is provided for backward compatibility because applications written for SQL Server version 7.0 or earlier may require the use of SQL Server logins and passwords. Additionally, SQL Server Authentication is required when an instance of SQL Server is running on Windows 98 because Windows Authentication Mode is not supported on Windows 98. Therefore, SQL Server uses Mixed Mode when running on Windows 98 (but supports only SQL Server Authentication).

Figure 6.2: SQL Server Security Decision Tree



Even though Windows Authentication is recommended, SQL Server Authentication may be required for connections with clients other than Windows NT 4.0 and Windows 2000 clients; it may also be necessary for legacy applications.

Notes

Notes When connecting to an instance of SQL Server running on Windows NT 4.0 or Windows 2000 using Named Pipes, the user must have permission to connect to the Windows NT Named Pipes IPC, \\<computername>**\IPC$**. If the user does not have permission to connect, it is not possible to connect to an instance of SQL Server using Named Pipes unless either the Windows NT 4.0 or Windows 2000 **guest** account on the computer is enabled (disabled by default), or the permission "access this computer from the network" is granted to their user account.

Self Assessment

Fill in the blanks:

1. A user passes through ..... stages of security when working in Microsoft® SQL Server.
2. The ..... stage controls the activities the user is allowed to perform in the SQL Server database.
3. ..... mode allows a user to connect through a Microsoft Windows NT® 4.0 or Windows® 2000 user account.
4. ..... allows users to connect to an instance of SQL Server using either Windows Authentication or SQL Server Authentication.
5. SQL Server Authentication is provided for ..... compatibility.
6. A user's network security attributes are established at ..... time and are validated by a Windows domain controller.
7. ..... integration operates over any supported network protocol in SQL Server.

6.2 Permissions

Every SQL Server securable has associated permissions that can be granted to a principal.

The following describes the general conventions that are followed for naming permissions:

- ***CONTROL:*** Confers ownership-like capabilities on the grantee. The grantee effectively has all defined permissions on the securable. A principal that has been granted CONTROL can also grant permissions on the securable. Because the SQL Server security model is hierarchical, CONTROL at a particular scope implicitly includes CONTROL on all the securables under that scope. For example, CONTROL on a database implies all permissions on the database, all permissions on all assemblies in the database, all permissions on all schemas in the database, and all permissions on objects within all schemas within the database.
- ***ALTER:*** Confers the ability to change the properties, except ownership, of a particular securable. When granted on a scope, ALTER also bestows the ability to alter, create, or drop any securable that is contained within that scope. For example, ALTER permission on a schema includes the ability to create, alter, and drop objects from the schema.
    - ALTER ANY <Server Securable>, where Server Securable can be any server securable.

      Confers the ability to create, alter, or drop individual instances of the Server Securable. For example, ALTER ANY LOGIN confers the ability to create, alter, or drop any login in the instance.

Notes

- ALTER ANY <Database Securable>, where Database Securable can be any securable at the database level.

  Confers the ability to CREATE, ALTER, or DROP individual instances of the Database Securable. For example, ALTER ANY SCHEMA confers the ability to create, alter, or drop any schema in the database.

- ***TAKE OWNERSHIP:*** Enables the grantee to take ownership of the securable on which it is granted.

  IMPERSONATE <Login>

  Enables the grantee to impersonate the login.

  IMPERSONATE <User>

  Enables the grantee to impersonate the user.

  CREATE <Server Securable>

  Confers to the grantee the ability to create the Server Securable.

  CREATE <Database Securable>

  Confers to the grantee the ability to create the Database Securable.

  CREATE <Schema-contained Securable>

  Confers the ability to create the schema-contained securable. However, ALTER permission on the schema is required to create the securable in a particular schema.

- ***VIEW DEFINITION:*** Enables the grantee to access metadata.
- ***REFERENCES:*** The REFERENCES permission on a table is needed to create a FOREIGN KEY constraint that references that table.

  The REFERENCES permission is needed on an object to create a FUNCTION or VIEW with the WITH SCHEMABINDING clause that references that object.

6.2.1 Permissions Applicable to Specific Securables

The following table lists major classes of permissions and the kinds of securables to which they may be applied.

Table 6.1

Permission	Applies to
SELECT	Synonyms Tables and columns Table-valued functions, Transact-SQL and common language runtime (CLR), and columns Views and columns
VIEW CHANGE TRACKING	Tables Schemas
UPDATE	Synonyms Tables and columns Views and columns
REFERENCES	Scalar and aggregate functions (Transact-SQL and CLR) Service Broker queues

Contd....

Notes

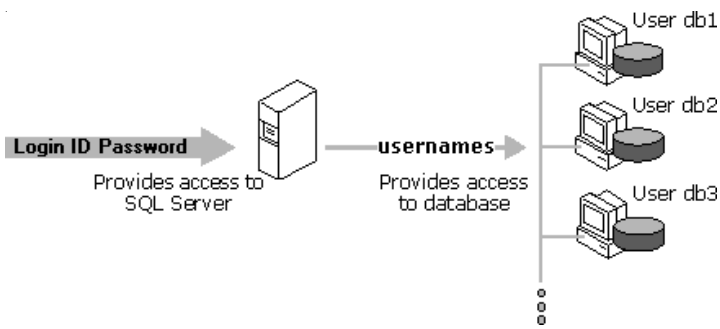
	Tables and columns Table-valued functions (Transact-SQL and CLR), and columns Types Views and columns
INSERT	Synonyms Tables and columns Views and columns
DELETE	Synonyms Tables and columns Views and columns
EXECUTE	Procedures (Transact-SQL and CLR) Scalar and aggregate functions (Transact-SQL and CLR) Synonyms CLR types
RECEIVE	Service Broker queues
VIEW DEFINITION	Procedures (Transact-SQL and CLR) Service Broker queues Scalar and aggregate functions (Transact-SQL and CLR) Synonyms Tables Table-valued functions (Transact-SQL and CLR) Views
ALTER	Procedures (Transact-SQL and CLR) Scalar and aggregate functions (Transact-SQL and CLR) Service Broker queues Tables Table-valued functions (Transact-SQL and CLR) Views
TAKE OWNERSHIP	Procedures (Transact-SQL and CLR) Scalar and aggregate functions (Transact-SQL and CLR) Synonyms Tables Table-valued functions (Transact-SQL and CLR) Views
CONTROL	Procedures (Transact-SQL and CLR) Scalar and aggregate functions (Transact-SQL and CLR) Service Broker queues Synonyms

6.2.2 Permissions Validation

After a user has been authenticated and allowed to log in to an instance of Microsoft® SQL Server™, a separate user account is required in each database the user must access. Requiring a user account in each database prevents users from connecting to an instance of SQL Server and accessing all the databases on a server. For example, if a server contains a **personnel** database and a **recruiting** database, users who should be able to access the **recruiting** database but not the **personnel** database would have a user account created only in the **recruiting** database.

Notes

Figure 6.3



The user account in each database is used to apply security permissions for the objects (for example, tables, views, and stored procedures) in that database. This user account can be mapped from Microsoft Windows NT® 4.0 and Windows® 2000 user accounts, Windows NT 4.0 and Windows 2000 groups in which the user is a member, or SQL Server login accounts. If there is no account mapped directly, the user may be allowed to work in a database under the **guest** account, if one exists. The activities a user is allowed to perform are controlled by the permissions applied to the user account from which they gained access to a database.

Notes **Security Note** When possible, use Windows Authentication. Also, avoid using the **guest** account; all logins without their own database permissions obtain the database permissions granted to this account. If you must use the **guest** account, grant it minimum permissions.

SQL Server accepts commands after a user gains access to a database. All activities a user performs in a database are communicated to SQL Server through Transact-SQL statements. When an instance of SQL Server receives a Transact-SQL statement, it ensures the user has permission to execute the statement in the database. If the user does not have permission to execute a statement or access an object used by the statement, SQL Server returns a permissions error.

6.3 Disadvantages of SQL Server Authentication

- If a user is a Windows domain user who has a login and password for Windows, he must still provide another (SQL Server) login and password to connect. Keeping track of multiple names and passwords is difficult for many users. Having to provide SQL Server credentials every time that one connects to the database can be annoying.
- SQL Server Authentication cannot use Kerberos security protocol.
- Windows offers additional password policies that are not available for SQL Server logins.

6.4 Advantages of SQL Server Authentication

- Allows SQL Server to support older applications and applications provided by third parties that require SQL Server Authentication.
- Allows SQL Server to support environments with mixed operating systems, where all users are not authenticated by a Windows domain.

Notes

- Allows users to connect from unknown or untrusted domains. For instance, an application where established customers connect with assigned SQL Server logins to receive the status of their orders.
- Allows SQL Server to support Web-based applications where users create their own identities.
- Allows software developers to distribute their applications by using a complex permission hierarchy based on known, preset SQL Server logins.

Notes Using SQL Server Authentication does not limit the permissions of local administrators on the computer where SQL Server is installed.

6.5 SQL Server Encryption

Encryption is the process of obfuscating data by the use of a key or password. This can make the data useless without the corresponding decryption key or password. Encryption does not solve access control problems. However, it enhances security by limiting data loss even if access controls are bypassed. For example, if the database host computer is misconfigured and a hacker obtains sensitive data, that stolen information might be useless if it is encrypted.

You can use encryption in SQL Server for connections, data, and stored procedures. The following table contains more information about encryption in SQL Server.

Important

Although encryption is a valuable tool to help ensure security, it should not be considered for all data or connections. When you are deciding whether to implement encryption, consider how users will access data. If users access data over a public network, data encryption might be required to increase security. However, if all access involves a secure intranet configuration, encryption might not be required. Any use of encryption should also include a maintenance strategy for passwords, keys, and certificates.

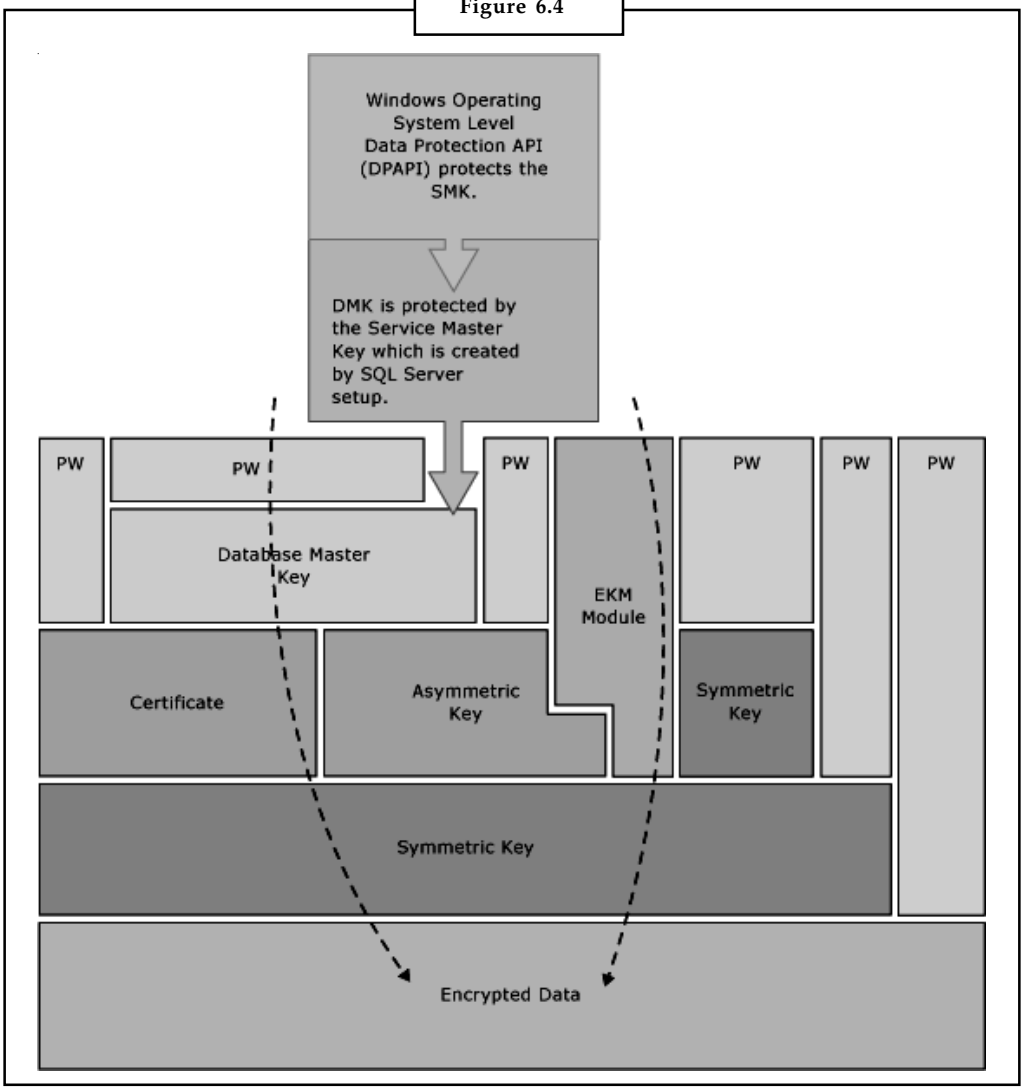
6.5.1 Encryption Hierarchy

SQL Server encrypts data with a hierarchical encryption and key management infrastructure. Each layer encrypts the layer below it by using a combination of certificates, asymmetric keys, and symmetric keys. Asymmetric keys and symmetric keys can be stored outside of SQL Server in an Extensible Key Management (EKM) module.

The following illustration shows that each layer of the encryption hierarchy encrypts the layer beneath it, and displays the most common encryption configurations. The access to the start of the hierarchy is usually protected by a password.

Notes

Figure 6.4



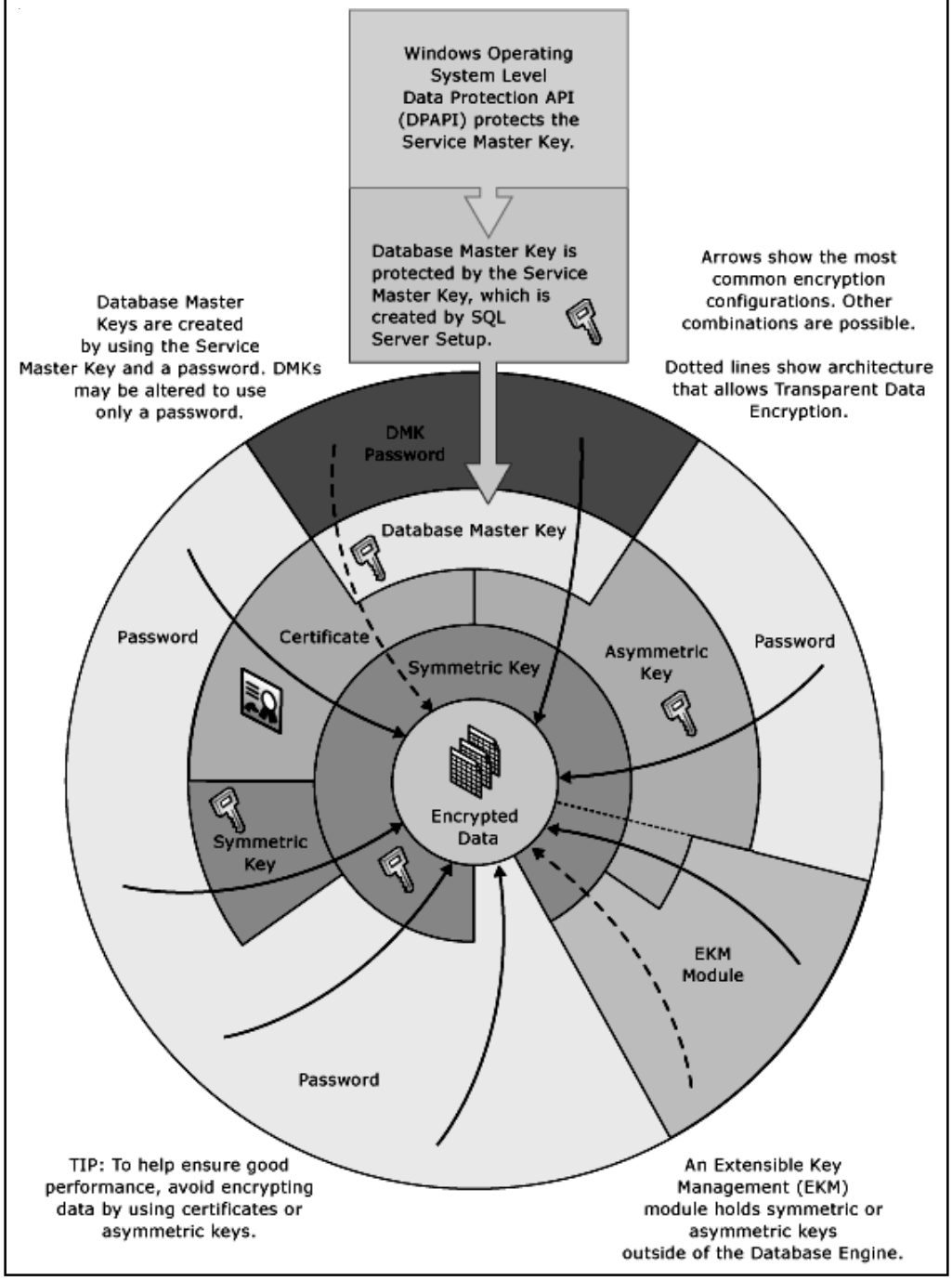
Keep in mind the following concepts:

- For best performance, encrypt data using symmetric keys instead of certificates or asymmetric keys.
- Database master keys are protected by the Service Master Key. The Service Master Key is created by SQL Server setup and is encrypted with the Windows Data Protection API (DPAPI).
- Other encryption hierarchies stacking additional layers are possible.
- An Extensible Key Management (EKM) module holds symmetric or asymmetric keys outside of SQL Server.
- Transparent Data Encryption (TDE) must use a symmetric key called the database encryption key which is protected by either a certificate protected by the database master key of the master database, or by an asymmetric key stored in an EKM.
- The Service Master Key and all Database Master Keys are symmetric keys.

Notes

The following illustration shows the same information in an alternative manner.

Figure 6.5



This diagram illustrates the following additional concepts:

- In this illustration, arrows indicate common encryption hierarchies.
- Symmetric and asymmetric keys in the EKM can protect access to the symmetric and asymmetric keys stored in SQL Server. The dotted line associated with EKM indicates that keys in the EKM could replace the symmetric and asymmetric keys stored in SQL Server.

Notes

6.5.2 Encryption Mechanisms

SQL Server provides the following mechanisms for encryption:

- Transact-SQL functions
- Asymmetric keys
- Symmetric keys
- Certificates
- Transparent Data Encryption

Transact-SQL Functions

Individual items can be encrypted as they are inserted or updated using Transact-SQL functions.

Certificates

A public key certificate, usually just called a certificate, is a digitally-signed statement that binds the value of a public key to the identity of the person, device, or service that holds the corresponding private key. Certificates are issued and signed by a certification authority (CA). The entity that receives a certificate from a CA is the subject of that certificate. Typically, certificates contain the following information.

- The public key of the subject.
- The identifier information of the subject, such as the name and e-mail address.
- The validity period. This is the length of time that the certificate is considered valid.

  A certificate is valid only for the period of time specified within it; every certificate contains **Valid From** and **Valid To** dates. These dates set the boundaries of the validity period. When the validity period for a certificate has passed, a new certificate must be requested by the subject of the now-expired certificate.
- Issuer identifier information.
- The digital signature of the issuer.

This signature attests to the validity of the binding between the public key and the identifier information of the subject. (The process of digitally signing information entails transforming the information, as well as some secret information held by the sender, into a tag called a signature.)

A primary benefit of certificates is that they relieve hosts of the need to maintain a set of passwords for individual subjects. Instead, the host merely establishes trust in a certificate issuer, which may then sign an unlimited number of certificates.

When a host, such as a secure Web server, designates an issuer as a trusted root authority, the host implicitly trusts the policies that the issuer has used to establish the bindings of certificates it issues. In effect, the host trusts that the issuer has verified the identity of the certificate subject. A host designates an issuer as a trusted root authority by putting the self-signed certificate of the issuer, which contains the public key of the issuer, into the trusted root certification authority certificate store of the host computer. Intermediate or subordinate certification authorities are trusted only if they have a valid certification path from a trusted root certification authority.

Notes

The issuer can revoke a certificate before it expires. Revocation cancels the binding of a public key to an identity that is asserted in the certificate. Each issuer maintains a certificate revocation list that can be used by programs when they are checking the validity of any given certificate.

The self-signed certificates created by SQL Server follow the X.509 standard and support the X.509 v1 fields.

Asymmetric Keys

An asymmetric key is made up of a private key and the corresponding public key. Each key can decrypt data encrypted by the other. Asymmetric encryption and decryption are relatively resource-intensive, but they provide a higher level of security than symmetric encryption. An asymmetric key can be used to encrypt a symmetric key for storage in a database.

Symmetric Keys

A symmetric key is one key that is used for both encryption and decryption. Encryption and decryption by using a symmetric key is fast, and suitable for routine use with sensitive data in the database.

Transparent Data Encryption

Transparent Data Encryption (TDE) is a special case of encryption using a symmetric key. TDE encrypts an entire database using that symmetric key called the database encryption key. The database encryption key is protected by other keys or certificates which are protected either by the database master key or by an asymmetric key stored in an EKM module.

Task List the various types of SQL server encryption mechanisms and differentiate between them.

6.6 Summary

- The authentication stage identifies the user using a login account and verifies only the ability to connect to an instance of SQL Server. If authentication is successful, the user connects to an instance of SQL Server.
- Microsoft SQL Server can operate in one of two security (authentication) modes:
  - Windows Authentication Mode (Windows Authentication): Windows Authentication mode allows a user to connect through a Microsoft Windows NT® 4.0 or Windows® 2000 user account.
  - Mixed Mode (Windows Authentication and SQL Server Authentication): Mixed Mode allows users to connect to an instance of SQL Server using either Windows Authentication or SQL Server Authentication. Users who connect through a Windows NT 4.0 or Windows 2000 user account can make use of trusted connections in either Windows Authentication Mode or Mixed Mode.
- Windows Authentication has certain benefits over SQL Server Authentication, primarily due to its integration with the Windows NT 4.0 and Windows 2000 security system. Windows NT 4.0 and Windows 2000 security provides more features, such as secure

Notes

validation and encryption of passwords, auditing, password expiration, minimum password length, and account lockout after multiple invalid login requests.

- SQL Server Authentication is provided for backward compatibility because applications written for SQL Server version 7.0 or earlier may require the use of SQL Server logins and passwords. Additionally, SQL Server Authentication is required when an instance of SQL Server is running on Windows 98 because Windows Authentication Mode is not supported on Windows 98. Therefore, SQL Server uses Mixed Mode when running on Windows 98 (but supports only SQL Server Authentication).
- After a user has been authenticated and allowed to log in to an instance of Microsoft® SQL Server™, a separate user account is required in each database the user must access. Requiring a user account in each database prevents users from connecting to an instance of SQL Server and accessing all the databases on a server.
- Encryption is the process of obfuscating data by the use of a key or password. This can make the data useless without the corresponding decryption key or password. Encryption does not solve access control problems. However, it enhances security by limiting data loss even if access controls are bypassed. For You can use encryption in SQL Server for connections, data, and stored procedures.
- SQL Server encrypts data with a hierarchical encryption and key management infrastructure. Each layer encrypts the layer below it by using a combination of certificates, asymmetric keys, and symmetric keys. Asymmetric keys and symmetric keys can be stored outside of SQL Server in an Extensible Key Management (EKM) module.

6.7 Keywords

Mixed Mode (Windows Authentication and SQL Server Authentication): Mixed Mode allows users to connect to an instance of SQL Server using either Windows Authentication or SQL Server Authentication.

SQL Server Authentication: The authentication stage identifies the user using a login account and verifies only the ability to connect to an instance of SQL Server. If authentication is successful, the user connects to an instance of SQL Server.

SQL Server Encryption: Encryption is the process of obfuscating data by the use of a key or password.

Windows Authentication Mode (Windows Authentication): Windows Authentication mode allows a user to connect through a Microsoft Windows NT® 4.0 or Windows® 2000 user account.

6.8 Review Questions

1. Describe the two main types of authentication modes in SQL server.
2. Differentiate between SQL server authentication and windows authentication
3. Discuss the permission validation process along with a suitable diagram.
4. Explain the advantages and disadvantages of SQL server authentication.
5. Define: SQL server encryption. Also explain its hierarchy along with the diagram.
6. Describe the various encryption mechanisms.
7. Comment: Although encryption is a valuable tool to help ensure security, it should not be considered for all data or connections.

Notes

Answers: Self Assessment

1. Two
2. permissions validation
3. Windows Authentication
4. Mixed Mode
5. Backward
6. network login
7. Login security

6.9 Further Readings

Books

C.J. Date, *Introduction to Database Systems*, Pearson Education.

Elmasri Navrate, *Fundamentals of Database Systems*, Pearson Education.

Peter Rob & Carlos Coronel, *Database Systems Design, Implementation and Management*, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, *Database Management Systems*, Tata McGraw Hill, 3rd Edition.

Silberschatz, Korth, *Database System Concepts*, McGraw Hill, 5th Edition.

Online links

www.sql-tutorial.net/

www.w3schools.com/sql/

www.sqlservertutorials.com/

Notes

Unit 7: Automating Administrating Tasks

CONTENTS

Objectives

Introduction

7.1 Meaning of Database Mail

7.1.1 Reliability Factors of Database Mail

7.1.2 Scalability Factors of Database Mail

7.1.3 Security Features of Database Mail

7.1.4 Supportability Features of Database Mail

7.1.5 Database Mail Configuration Wizard

7.1.6 Database Mail Architecture

7.2 Event Notifications

7.2.1 Understanding Event Notifications

7.2.2 Designing Event Notifications

7.2.3 Defining Notification Scope

7.2.4 Creating the Event Notification

7.3 SQL Server Agent

7.3.1 Meaning of SQL Server Agent

7.3.2 Configuring SQL Server Agent

7.3.3 Setting Required Permissions

7.4 Maintenance Plans

7.4.1 Features

7.4.2 How to Create a Maintenance Plan?

7.4.3 Maintenance Plan Wizard

7.5 Summary

7.6 Keywords

7.7 Review Questions

7.8 Further Readings

Objectives

After studying this unit, you will be able to:

- Know about various types of automating administrative tasks in SQL server
- Comprehend the meaning and features of database mail
- Understand the concept of SQL server agent
- Familiarize with maintenance plans in SQL server

Notes

Introduction

Microsoft SQL Server allows you to automate administrative tasks. To automate administration, you define predictable administrative tasks and then specify the conditions under which each task occurs. Using automated administration to handle routine tasks and events frees your time to perform other administrative functions.

7.1 Meaning of Database Mail

Database Mail is an enterprise solution for sending e-mail messages from the SQL Server Database Engine. Using Database Mail, your database applications can send e-mail messages to users. The messages can contain query results, and can also include files from any resource on your network. Database Mail is designed for reliability, scalability, security, and supportability.

Notes Database Mail is not active by default. To use Database Mail, you must explicitly enable Database Mail by using either the Database Mail Configuration Wizard, the sp_configure stored procedure, or by using the Surface Area Configuration facet of Policy-Based Management..

7.1.1 Reliability Factors of Database Mail

- No Microsoft Outlook or Extended Messaging Application Programming Interface (Extended MAPI) requirement. Database Mail uses the standard Simple Mail Transfer Protocol (SMTP) to send mail. You can use Database Mail without installing an Extended MAPI client on the computer that runs SQL Server.
- Process isolation. To minimize the impact on SQL Server, the component that delivers e-mail runs outside of SQL Server, in a separate process. SQL Server will continue to queue e-mail messages even if the external process stops or fails. The queued messages will be sent once the outside process or SMTP server comes online.
- Failover accounts. A Database Mail profile allows you to specify more than one SMTP server. Should an SMTP server be unavailable, mail can still be delivered to another SMTP server.
- Cluster support. Database Mail is cluster-aware and is fully supported on a cluster.

7.1.2 Scalability Factors of Database Mail

- ***Background delivery.*** Database Mail provides background, or asynchronous, delivery. When you call **sp_send_dbmail** to send a message, Database Mail adds a request to a Service Broker queue. The stored procedure returns immediately. The external e-mail component receives the request and delivers the e-mail.
- ***Multiple profiles.*** Database Mail allows you to create multiple profiles within a SQL Server instance. Optionally, you can choose the profile that Database Mail uses when you send a message.
- ***Multiple accounts.*** Each profile can contain multiple failover accounts. You can configure different profiles with different accounts to distribute e-mail across multiple e-mail servers.
- ***64-bit compatibility.*** Database Mail is fully supported on 64-bit installations of SQL Server.

Notes

7.1.3 Security Features of Database Mail

- ***Off by default.*** To reduce the surface area of SQL Server, Database Mail stored procedures are disabled by default.
- ***To send Database Mail***, you must be a member of the **DatabaseMailUserRole** database role in the **msdb** database.
- ***Profile security.*** Database Mail enforces security for mail profiles. You choose the **msdb** database users or groups that have access to a Database Mail profile. You can grant access to either specific users, or all users in **msdb**. A private profile restricts access to a specified list of users. A public profile is available to all users in a database.
- ***Attachment size governor.*** Database Mail enforces a configurable limit on the attachment file size. You can change this limit by using the sysmail_configure_sp stored procedure.
- ***Prohibited file extensions.*** Database Mail maintains a list of prohibited file extensions. Users cannot attach files with an extension that appears in the list. You can change this list by using sysmail_configure_sp.
- ***Database Mail runs under the SQL Server Engine service account.*** To attach a file from a folder to an email, the SQL Server engine account should have permissions to access the folder with the file.

7.1.4 Supportability Features of Database Mail

- ***Integrated configuration.*** Database Mail maintains the information for e-mail accounts within SQL Server Database Engine. There is no need to manage a mail profile in an external client application. Database Mail Configuration Wizard provides a convenient interface for configuring Database Mail. You can also create and maintain Database Mail configurations using Transact-SQL.
- ***Logging.*** Database Mail logs e-mail activity to SQL Server, the Microsoft Windows application event log, and to tables in the **msdb** database.
- ***Auditing.*** Database Mail keeps copies of messages and attachments sent in the **msdb** database. You can easily audit Database Mail usage and review the retained messages.
- ***Support for HTML.*** Database Mail allows you to send e-mail formatted as HTML.

Database Mail provides a robust, high-performance replacement for the most commonly requested features of SQL Mail. Database Mail is designed to operate with SMTP servers, and is tested with Microsoft SMTP servers.

Notes Database Mail is not available in SQL Server Express.

7.1.5 Database Mail Configuration Wizard

The Database Mail Configuration Wizard provides a convenient way to manage Database Mail configuration objects and enables Database Mail, if needed. To use this wizard, you must be a member of the **sysadmin** fixed server role. To send Database Mail, you must be a member of the **DatabaseMailUserRole** database role in the msdb database.

Notes

To start the Database Mail Configuration Wizard

1. From Object Explorer, connect to an instance of SQL Server.
2. Expand Management, right-click Database Mail, and select Configure Database Mail.
3. Choose the Set up Database Mail option to set up Database Mail for the first time.

> *Notes* Enabling SQL Server Service Broker in any database requires a database lock. If Service Broker was deactivated in **msdb**, to enable Database Mail, first stop SQL Server Agent so that Service Broker can obtain the necessary lock.

4. Choose one of the other options for specific maintenance tasks:
   i. Manage Database Mail accounts and profiles.
   ii. Manage profile security.
   iii. View or change system parameters.

7.1.6 Database Mail Architecture

Database Mail consists of the following main components:

1. ***Configuration and security components:*** Database Mail stores configuration and security information in the **msdb** database. Configuration and security objects create profiles and accounts used by Database Mail.
2. ***Messaging components:*** The **msdb** database acts as the mail-host database that holds the messaging objects that Database Mail uses to send e-mail. These objects include the **sp_send_dbmail** stored procedure and the data structures that hold information about messages.
3. ***Database mail executable:*** The Database Mail executable is an external program that reads from a queue in the **msdb** database and sends messages to e-mail servers.
4. ***Logging and auditing components:*** Database Mail records logging information in the **msdb** database and the Microsoft Windows application event log.

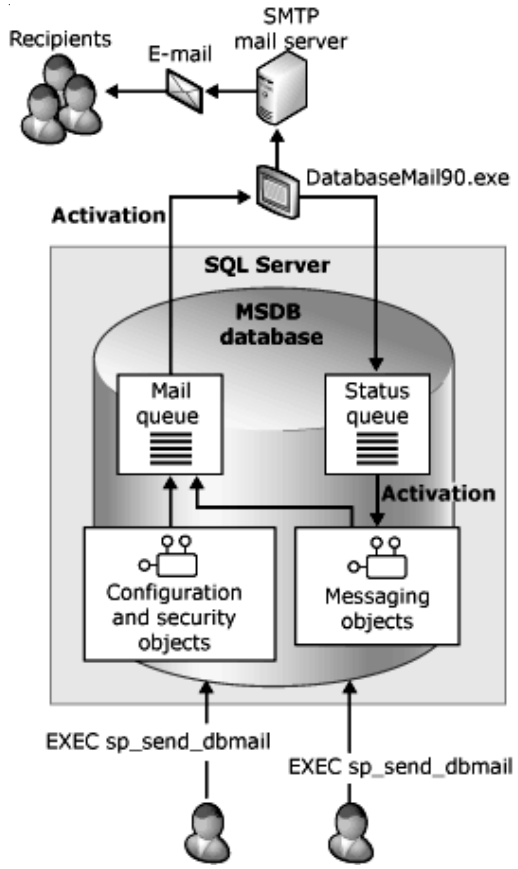
You do not have to have a complete understanding of the Database Mail architecture to use Database Mail effectively. However, understanding the Database Mail components and how the components interact might help you design your applications and troubleshoot any problems that may occur.

The following Figure 7.1 shows an overview of the Database Mail architecture.

Database Mail is designed on a queued architecture that uses service broker technologies. When users execute **sp_send_dbmail**, the stored procedure inserts an item into the mail queue and creates a record that contains the e-mail message. Inserting the new entry in the mail queue starts the external Database Mail process (DatabaseMail.exe). The external process reads the e-mail information and sends the e-mail message to the appropriate e-mail server or servers. The external process inserts an item in the Status queue for the outcome of the send operation. Inserting the new entry in the status queue starts an internal stored procedure that updates the status of the e-mail message. Besides storing the sent, or unsent, e-mail message, Database Mail also records any e-mail attachments in the system tables. Database Mail views provide the status of messages for troubleshooting, and stored procedures allow for administration of the Database Mail queue.

Notes

Figure 7.1: Database Mail Architecture



Only members of the **DatabaseMailUserRole** in the **msdb** database can execute **sp_send_dbmail**.

Self Assessment

Fill in the blanks:

1. Database Mail is designed on a .....architecture that uses service broker technologies.
2. Configuration and security objects create profiles and accounts used by .....
3. To send Database Mail, you must be a member of the ..... database role in the msdb database.
4. Database Mail uses the standard ..... to send mail.
5. Database Mail is designed for reliability, scalability, security, and .....

7.2 Event Notifications

Event notifications are a special kind of database object and send information about server and database events to a Service Broker service. This section provides the information that is required to understand, design, and implement event notifications.

Notes

7.2.1 Understanding Event Notifications

Event notifications execute in response to a variety of Transact-SQL data definition language (DDL) statements and SQL Trace events by sending information about these events to a Service Broker service.

Event notifications can be used to do the following:

- Log and review changes or activity occurring on the database.
- Perform an action in response to an event in an asynchronous instead of synchronous manner.

Event notifications can offer a programming alternative to DDL triggers and SQL Trace.

Event notifications run asynchronously, outside the scope of a transaction. Therefore, unlike DDL triggers, event notifications can be used inside a database application to respond to events without using any resources defined by the immediate transaction.

Unlike SQL Trace, event notifications can be used to perform an action inside an instance of SQL Server in response to a SQL Trace event.

When an event notification is created, one or more Service Broker conversations between an instance of SQL Server and the target service you specify are opened. The conversations typically remain open as long as the event notification exists as an object on the server instance. In some error cases the conversations can close before the event notification is dropped. These conversations are never shared between event notifications. Every event notification has its own exclusive conversations. Ending a conversation explicitly prevents the target service from receiving more messages, and the conversation will not reopen the next time the event notification fires.

Event information is delivered to the Service Broker as a variable of type **xml** that provides information about when an event occurs, about the database object affected, the Transact-SQL batch statement involved, and other information.

Event data can be used by applications that are running together with SQL Server to track progress and make decisions. For example, the following event notification sends a notice to a certain service every time an ALTER TABLE statement is issued in the **AdventureWorks** sample database.

```
USE AdventureWorks
GO
CREATE EVENT NOTIFICATION NotifyALTER_T1
ON DATABASE
FOR ALTER_TABLE
TO SERVICE '//Adventure-Works.com/ArchiveService' ,
        '8140a771-3c4b-4479-8ac0-81008ab17984';
```

7.2.2 Designing Event Notifications

To design an event notification, you must determine the following:

- The scope of the notification.
- The Transact-SQL statement, or group of statements, that raises the event notification.

Notes

7.2.3 Defining Notification Scope

You can specify an event notification to occur in response to a statement made on all objects in the current database or all objects on an instance of SQL Server. Event notifications specified on the QUEUE_ACTIVATION and BROKER_QUEUE_DISABLED events are scoped to individual queues. Not all events can occur at any scope. CREATE_DATABASE events, for example, can occur only at the server instance level. Conversely, an event notification created on an ALTER_TABLE event can be programmed to occur on all tables in the database or on all tables on the server instance.

The following example sends a notification of any ALTER TABLE statement run on the server instance to the Service Broker instance in the current database.

```
CREATE EVENT NOTIFICATION log_ddl1
        ON SERVER
        FOR ALTER_TABLE
        TO SERVICE '//Adventure-Works.com/ArchiveService' , 'current database';
```

Links to the Transact-SQL statements and the scopes that can be specified for them are provided in the section "Selecting a Particular DDL Statement to Raise an Event Notification" below.

Event notifications do not occur in response to events that affect local or global temporary tables and stored procedures.

Specifying a Transact-SQL Statement or Group of Statements

Event notifications can be created to occur in response to the following:

- A particular DDL statement, SQL Trace event, or Service Broker event
- A predefined group of DDL statements or SQL Trace events

Selecting a Particular DDL Statement to Raise an Event Notification

Event notifications can be designed to occur after a particular Transact-SQL statement or stored procedure is run. As shown in the previous example, that event notification occurs after an ALTER_TABLE event.

A list of the individual Transact-SQL statements that can be specified to raise an event notification and the scope at which the notifications can execute is provided in DDL Statements for Use with Event Notifications. These events can also be obtained by querying the sys.event_notification_event_types catalog view.

Notes Certain system stored procedures that perform DDL-like operations can also fire event notifications. Test your event notifications to determine their responses to system stored procedures that are run. For example, the CREATE TYPE statement and the **sp_addtype** stored procedure will both fire an event notification that is created on a CREATE_TYPE event. However, the **sp_rename** stored procedure does not fire any event notifications.

Selecting a Particular SQL Trace Event to Raise an Event Notification

Event notifications can be designed to fire after a SQL Trace event occurs. For example, the following event notification fires after an Object_Created event on the server.

```
CREATE EVENT NOTIFICATION log_ddl1
```

Notes

```
ON SERVER
FOR Object_Created
TO SERVICE '//Adventure-Works.com/ArchiveService', 'current database' ;
```

SQL Trace events can be executed only at the server instance scope.

Selecting a Service Broker Event to Raise an Event Notification

Event notifications can be designed to fire after a QUEUE_ACTIVATION or BROKER_QUEUE_DISABLED Service Broker event. The QUEUE_ACTIVATION event occurs when a queue has messages to process. The BROKER_QUEUE_DISABLED event occurs when the status of a queue is set to OFF.

Selecting a Predefined Group of DDL Statements to Raise an Event Notification

An event notification can occur after any Transact-SQL event that belongs to a predefined grouping of similar events is run. For example, if you want an event notification to occur after any CREATE TABLE, ALTER TABLE, or DROP TABLE statement is executed, you can specify FOR DDL_TABLE_EVENTS in the CREATE EVENT NOTIFICATION statement. After CREATE EVENT NOTIFICATION executes, the events that are covered by an event group are added to the **sys.events** catalog view.

For a list of the predefined groups of DDL and DML statements that are available for event notifications, the particular statements they cover, and the scope at which these event groups can execute, see DDL Event Groups for Use with Event Notifications.

Selecting a Predefined Group of SQL Trace Events to Raise an Event Notification

An event notification can occur after any SQL Trace event that belongs to a predefined grouping of similar trace events is run. For example, if you want an event notification to occur after any locking-related trace event, which includes the LOCK_DEADLOCK, LOCK_DEADLOCK_CHAIN, LOCK_ESCALATION, and DEADLOCK_GRAPH events, you can specify FOR TRC_LOCKS in the CREATE EVENT NOTIFICATION statement.

For a list of the predefined groups of SQL Trace events that are available for event notifications, see Trace Event Groups for Use with Event Notifications. These groups can execute only at the server instance level.

Important

Service Broker dialog security should be configured for event notifications that send messages to a service broker on a remote server. Dialog security must be configured manually according to the full security model.

Creating the Target Service

You do not have to create a Service Broker-initiating service because Service Broker includes the following specific message type and contract for event notifications:

http://schemas.microsoft.com/SQL/Notifications/PostEventNotification

The target service that receives event notifications must honor this preexisting contract.

Notes

To create a target service:

1. Create a queue to receive messages.

> *Notes* The queue receives the following message type: http://schemas.microsoft.com/SQL/Notifications/QueryNotification.

2. Create a service on the queue that references the event notifications contract.
3. Create a route on the service to define the address to which Service Broker sends messages for the service. For event notifications that target a service in the same database, specify ADDRESS = 'LOCAL'.

> *Notes* Service Broker routing determines the service that receives the notification messages. If the event notification targets a service on a remote server, both the source server and the target server must have routes defined on them to make sure that two-way communication occurs.

The following example creates a queue, a service on the queue, and a route on the service to handle messages from the event notification contract.

```
CREATE QUEUE NotifyQueue;
GO
CREATE SERVICE NotifyService
ON QUEUE NotifyQueue
(
[http://schemas.microsoft.com/SQL/Notifications/PostEventNotification]
);
GO
CREATE ROUTE NotifyRoute
WITH SERVICE_NAME = 'NotifyService',
ADDRESS = 'LOCAL';
GO
```

7.2.4 Creating the Event Notification

Event notifications are created by using the Transact-SQL CREATE EVENT NOTIFICATION statement, and are dropped by using the DROP EVENT NOTIFICATION STATEMENT. To modify an event notification, you must drop and re-create the event notification.

The following example creates the event notification CreateDatabaseNotification. This notification sends a message about any CREATE_DATABASE event that occurs on the server to the NotifyService service that was previously created.

Notes

```
CREATE EVENT NOTIFICATION CreateDatabaseNotification
ON SERVER
FOR CREATE_DATABASE
TO SERVICE 'NotifyService', '8140a771-3c4b-4479-8ac0-81008ab17984' ;
```

Caution Event notifications recognize CREATE_SCHEMA events and the <schema_element> definitions of CREATE SCHEMA statements as separate events. For example, an event notification is created on both the CREATE_SCHEMA and CREATE_TABLE events, and you run the following batch. CREATE SCHEMA CREATE TABLE t1 (col1 int) In this case, the event notification is raised two times: One time when the CREATE_SCHEMA event occurs, and again when the CREATE_TABLE event occurs. We recommend that you either avoid creating event notifications on both the CREATE_SCHEMA events and the <schema_element> texts of any corresponding CREATE SCHEMA definitions, or build logic into your application to avoid capturing unwanted event data.

To create an event notification

- CREATE EVENT NOTIFICATION (Transact-SQL)

To drop an event notification

- DROP EVENT NOTIFICATION (Transact-SQL)

Self Assessment

State true or false:

6. To modify an event notification, you must drop and recreate the event notification.
7. Service Broker routing determines the service that receives the notification messages.
8. An event notification can occur after any Transact-SQL event that belongs to a predefined grouping of similar events is run.
9. Event notifications are a special kind of database functions and send information about server and database events to a Service Broker service.
10. Event notifications recognize CREATE_SCHEMA events and the <schema_element> definitions of CREATE SCHEMA statements as same events.

7.3 SQL Server Agent

7.3.1 Meaning of SQL Server Agent

SQL Server Agent is a Microsoft Windows service that allows you to automate some administrative tasks. SQL Server Agent runs jobs, monitors SQL Server, and processes alerts. The SQL Server Agent service must be running before local or multiserver administrative jobs can run automatically.

7.3.2 Configuring SQL Server Agent

You can specify some configuration options for SQL Server Agent during installation of SQL Server. The full set of SQL Server Agent configuration options is only available within SQL

Notes

Server Management Studio, SQL Server Management Objects (SMO), or the SQL Server Agent stored procedures.

Notes Click SQL Server Agent in Object Explorer of SQL Server Management Studio to administer jobs, operators, alerts, and the SQL Server Agent service. However, Object Explorer only displays the SQL Server Agent node if you have permission to use it.

Caution Auto-restart should not be enabled for the SQL Server service or the SQL Server Agent service on failover cluster instances.

SQL Server Agent stores most configuration information in tables located in the **msdb** database. SQL Server Agent uses SQL Server credential objects to store the authentication information for proxies.

7.3.3 Setting Required Permissions

To perform its functions, SQL Server Agent must be configured to use the credentials of an account that is a member of the **sysadmin** fixed server role in SQL Server. The account must have the following Windows permissions:

- Log on as a service (SeServiceLogonRight)
- Replace a process-level token (SeAssignPrimaryTokenPrivilege)
- Bypass traverse checking (SeChangeNotifyPrivilege)
- Adjust memory quotas for a process (SeIncreaseQuotaPrivilege)

To verify that each of these required Windows permissions is set

1. Click Start, click Control Panel, Administrative Tools, and Local Security Policy.
2. Expand the Local Policies folder, and then click the User Rights Assignment folder.
3. Repeat the following steps for each permission:
   i. Right-click a permission (such as Log on as a service), and then click Properties.
   ii. In the properties dialog box (for example Log on as a service Properties), verify that the account under which SQL Server Agent runs is listed.
   iii. If it is not listed, click Add User or Group, enter the account under which SQL Server Agent runs, and then click OK.

Typically, the account selected for the SQL Server Agent is a domain account created for that purpose and has tightly controlled access permissions. It is not necessary to use a domain account, but if you use an account on the local computer, SQL Server Agent will not have permission to access resources on other computers. It is common for SQL Server to need permission on other computers, for instance when it creates a database backup and stores the file in a location on another computer.

Notes

7.4 Maintenance Plans

In earlier versions of SQL Server, you create database maintenance plans by using a single wizard, and they create multiple agent jobs as an output. These agent jobs themselves contain command-line parameters that indicate the function to perform. Maintenance plans create a workflow of the tasks required to make sure that your database is optimized, is regularly backed up, and is free of inconsistencies. The Maintenance Plan Wizard also creates core maintenance plans, but creating plans manually gives you much more flexibility. Maintenance plans create an Integration Services package, which is run by a SQL Server Agent job. These maintenance tasks can be run manually or automatically at scheduled intervals.

7.4.1 Features

Maintenance plans provide the following features:

- Workflow creation using a variety of typical maintenance tasks. You can also create your own custom Transact-SQL scripts.
- Conceptual hierarchies. Each plan lets you create or edit task workflows. Tasks in each plan can be grouped into subplans, which can be scheduled to run at different times.
- Support for multiserver plans that can be used in master server/target server environments.
- Support for logging plan history to remote servers.
- Support for Windows Authentication and SQL Server Authentication. When possible, use Windows Authentication.

Maintenance plans only run against databases set to compatibility level 80 or higher. The maintenance plan designer in SQL Server Management Studio does not display databases set to compatibility level 70 or lower.

You can migrate database maintenance plans by right-clicking the database maintenance plan and choosing Migrate.

You must be a member of the sysadmin role to create and manage maintenance plans, and to view them in Object Explorer. The SQL Server Agent node in Object Explorer is only displayed for members of the sysadmin fixed server role, SQLAgentReaderRole, SQLAgentUserRole, or SQLAgentOperatorRole fixed database roles.

Did u know? Members of the db_ssisadmin role and the dc_admin role may be able to elevate their privileges to sysadmin. This elevation of privilege can occur because these roles can modify Integration Services packages and Integration Services packages can be executed by SQL Server using the sysadmin security context of SQL Server Agent. To guard against this elevation of privilege when running maintenance plans, data collection sets, and other Integration Services packages, configure SQL Server Agent jobs that run packages to use a proxy account with limited privileges or only add sysadmin members to the db_ssisadmin and dc_admin roles.

7.4.2 How to Create a Maintenance Plan?

There are two ways to create a maintenance plan: you can create a plan using the Maintenance Plan Wizard, or you can create a plan using the design surface. The Wizard is best for creating basic maintenance plans, while creating a plan using the design surface allows you to utilize enhanced workflow.

Notes

To create or manage Maintenance Plans, you must be a member of the **sysadmin** fixed server role. Note that Object Explorer only displays maintenance plans if the user is a member of the **sysadmin** fixed server role.

To create a maintenance plan using the Maintenance Plan Wizard

1. In Object Explorer, expand a server, and then expand Management.
2. Right-click Maintenance Plans and select Maintenance Plan Wizard.
3. Follow the steps of the wizard to create a maintenance plan.

To create a maintenance plan using the design surface

1. In Object Explorer, expand a server, and then expand Management.
2. Right-click Maintenance Plans and select New Maintenance Plan.
3. In the New Maintenance Plan dialog box, type a name for the plan.
4. The Toolbox opens, and the <Maintenance Plan Name> [Design] surface opens with the default Subplan_1created. Optionally enter a description for the entire plan on the Design tab.
5. Double-click Subplan_1, and enter a name and description for the subplan in the Subplan Properties dialog box. Click the Subplan Schedule icon to enter schedule details in the Job Schedule Properties dialog box.
6. To build the subplan, drag and drop task flow elements from the Toolbox to the plan design surface to define the tasks that will be performed. Double-click tasks to open dialog boxes to configure the task options.
7. To define the workflow between tasks, click the task you want to occur first, hold the CTRL key down, and then click the task you want to occur second. Right-click either task, and then click Add Precedence Constraint.
8. To add another subplan that contains tasks run on a different schedule, click Add Subplan on the toolbar.
9. To add connections to different servers, click Manage Connections.
10. To specify reporting options, click the Reporting and Logging icon. To save more detailed information, select Log extended information. To write maintenance plan results information to another server, select Log to remote server. To view the results in the log file viewer, right-click the Maintenance Plans node or the specific maintenance plan, and then click

7.4.3 Maintenance Plan Wizard

The Maintenance Plan Wizard helps you set up the core maintenance tasks to make sure that your database performs well, is regularly backed up, and is free of inconsistencies. The maintenance plan wizard creates one or more SQL Server Agent jobs that perform these tasks on local servers or on target servers in a multiserver environment. Execution can be at scheduled intervals or on demand.

To create or manage maintenance plans, you must be a member of the **sysadmin** fixed server role. Note that Object Explorer only displays maintenance plans if the user is a member of the **sysadmin** fixed server role.

Notes

Maintenance plans can be created to perform the following tasks:

- Reorganize the data on the data and index pages by rebuilding indexes with a new fill factor. Rebuilding indexes with a new fill factor makes sure that database pages contain an equally distributed amount of data and free space. It also enables faster growth in the future.
- Compress data files by removing empty database pages.
- Update index statistics to make sure the query optimizer has current information about the distribution of data values in the tables. This enables the query optimizer to make better judgments about the best way to access data, because it has more information about the data stored in the database. Although index statistics are automatically updated by SQL Server periodically, this option can force the statistics to update immediately.
- Perform internal consistency checks of the data and data pages within the database to make sure that a system or software problem has not damaged data.
- Back up the database and transaction log files. Database and log backups can be retained for a specified period. This lets you create a history of backups to be used if you have to restore the database to a time earlier than the last database backup. You can also perform differential backups.
- Run SQL Server Agent jobs. This can be used to create jobs that perform a variety of actions, and also the maintenance plans to run the jobs.

The results generated by the maintenance tasks can be written as a report to a text file, or written to the maintenance plan tables, **sysmaintplan_log** and **sysmaintplan_logdetail**, in **msdb**. To view the results in the log file viewer, right-click Maintenance Plans, and then click View History.

Task Create a maintenance plan using the maintenance plan wizard for reorganizing the data and index pages by rebuilding indexes with a new fill factor.

Maintenance plans only run against databases set to compatibility level 80 or higher. The maintenance plan wizard does not display databases set to compatibility level 70 or lower.

7.5 Summary

- Microsoft SQL Server allows you to automate administrative tasks. Using automated administration to handle routine tasks and events frees your time to perform other administrative functions.
- Database Mail provides a robust, high-performance replacement for the most commonly requested features of SQL Mail. Database Mail is designed to operate with SMTP servers, and is tested with Microsoft SMTP servers.
- Event notifications execute in response to a variety of Transact-SQL data definition language (DDL) statements and SQL Trace events by sending information about these events to a Service Broker service.
- SQL Server Agent is a Microsoft Windows service that allows you to automate some administrative tasks. SQL Server Agent runs jobs, monitors SQL Server, and processes

Notes

alerts. The SQL Server Agent service must be running before local or multiserver administrative jobs can run automatically.

- Maintenance plans create a workflow of the tasks required to make sure that your database is optimized, is regularly backed up, and is free of inconsistencies. The Maintenance Plan Wizard also creates core maintenance plans, but creating plans manually gives you much more flexibility. Maintenance plans create an Integration Services package, which is run by a SQL Server Agent job.

7.6 Keywords

Database Mail: Database Mail is an enterprise solution for sending e-mail messages from the SQL Server Database Engine.

Event Notifications: Event notifications are a special kind of database object and send information about server and database events to a Service Broker service.

SQL Server Agent: SQL Server Agent is a Microsoft Windows service that allows you to automate some administrative tasks.

7.7 Review Questions

1. What are the various types of automating administrative tasks in SQL server
2. What do you mean by a database mail?
3. Explain the basic features of database mail.
4. Discuss the concept of SQL server agent.
5. Describe the concept of maintenance plans in SQL server.
6. Illustrate the concept of database mail architecture.
7. Discuss the steps required to start the database mail configuration wizard.
8. What are event notifications? Discuss the use of event notifications.

Answers: Self Assessment

1.	Queued	2.	Database Mail
3.	DatabaseMailUserRole	4.	Simple Mail Transfer Protocol (SMTP)
5.	Supportability	6.	True
7.	True	8.	True
9.	False	10.	False

7.8 Further Readings

Books

C.J. Date, *Introduction to Database Systems*, Pearson Education.

Elmasri Navrate, *Fundamentals of Database Systems*, Pearson Education.

Notes

Peter Rob & Carlos Coronel, *Database Systems Design, Implementation and Management*, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, *Database Management Systems*, Tata McGraw Hill, 3rd Edition.

Silberschatz, Korth, *Database System Concepts*, McGraw Hill, 5th Edition.

Online links

www.sql-tutorial.net/

www.w3schools.com/sql/

www.sqlservertutorials.com/

Notes

Unit 8: Configuring SQL Server Network Communication

CONTENTS

Objectives

Introduction

8.1 Network Protocols

8.1.1 Shared Memory

8.1.2 TCP/IP

8.1.3 Named Pipes

8.1.4 VIA

8.2 Enabling and Configuring Protocols

8.3 Native Client Configuration

8.3.1 Client Protocols Properties (Order Tab)

8.3.2 Client Protocols - Shared Memory Properties (Protocol Tab)

8.3.3 Client Protocols - TCP/IP Properties (Protocol Tab)

8.3.4 Client Protocols - Named Pipes Properties (Protocol Tab)

8.3.5 Client Protocols - VIA Properties (Protocol Tab)

8.4 Summary

8.5 Keywords

8.6 Review Questions

8.7 Further Readings

Objectives

After studying this unit, you will be able to:

- Learn about different types of network protocols used in SQL server
- Understand the process to configure and secure the network protocols
- Explain the use and functions of SQL Configuration Manager
- Know how to configure native client configuration

Introduction

Server network configuration tasks include enabling protocols, modifying the port or pipe used by a protocol, configuring encryption, configuring the SQL Server Browser service, exposing or hiding the Microsoft Database Engine on the network, and registering the Server Principal Name. Most of the time, you do not need to change the server network configuration. Only reconfigure the server network protocols if special network requirements.

Network configuration for SQL Server 2005 is done using SQL Server Configuration Manager.

Notes

8.1 Network Protocols

SQL Server provides support for four network protocols.

To connect to SQL Server Database Engine you must have a network protocol enabled. Microsoft SQL Server can service requests on several protocols at the same time. Clients connect to SQL Server with a single protocol. If the client program does not know which protocol SQL Server is listening on, configure the client to sequentially try multiple protocols. Use SQL Server Configuration Manager to enable, disable, and configure network protocols.

8.1.1 Shared Memory

Shared memory is the simplest protocol to use and has no configurable settings. Because clients using the shared memory protocol can only connect to a SQL Server instance running on the same computer, it is not useful for most database activity. Use the shared memory protocol for troubleshooting when you suspect the other protocols are configured incorrectly.

Notes Clients that use MDAC 2.8 or earlier cannot use shared memory protocol. If these clients try this, they are automatically switched to the named pipes protocol.

8.1.2 TCP/IP

TCP/IP is a common protocol widely used over the Internet. It communicates across interconnected networks of computers that have diverse hardware architectures and various operating systems. TCP/IP includes standards for routing network traffic and offers advanced security features. It is the most popular protocol that is used in business today. Configuring your computer to use TCP/IP can be complex, but most networked computers are already correctly configured.

8.1.3 Named Pipes

Named Pipes is a protocol developed for local area networks. A part of memory is used by one process to pass information to another process, so that the output of one is the input of the other. The second process can be local (on the same computer as the first) or remote (on a networked computer).

8.1.4 VIA

Virtual Interface Adapter (VIA) protocol works with VIA hardware.

Caution The VIA protocol is deprecated. This feature will be removed in a future version of Microsoft SQL Server. Avoid using this feature in new development work, and plan to modify applications that currently use this feature.

Named Pipes vs. TCP/IP Sockets

In a fast local area network (LAN) environment, Transmission Control Protocol/Internet Protocol (TCP/IP) Sockets and Named Pipes clients are comparable with regard to performance. However, the performance difference between the TCP/IP Sockets and Named Pipes clients becomes

Notes

apparent with slower networks, such as across wide area networks (WANs) or dial-up networks. This is because of the different ways the interprocess communication (IPC) mechanisms communicate between peers.

For named pipes, network communications are typically more interactive. A peer does not send data until another peer asks for it using a read command. A network read typically involves a series of peek named pipes messages before it starts to read the data. These can be very costly in a slow network and cause excessive network traffic, which in turn affects other network clients.

It is also important to clarify if you are talking about local pipes or network pipes. If the server application is running locally on the computer that is running an instance of SQL Server, the local Named Pipes protocol is an option. Local named pipes runs in kernel mode and is very fast.

For TCP/IP Sockets, data transmissions are more streamlined and have less overhead. Data transmissions can also take advantage of TCP/IP Sockets performance enhancement mechanisms such as windowing, delayed acknowledgements, and so on. This can be very helpful in a slow network. Depending on the type of applications, such performance differences can be significant.

TCP/IP Sockets also support a backlog queue. This can provide a limited smoothing effect compared to named pipes that could lead to pipe-busy errors when you are trying to connect to SQL Server.

Generally, TCP/IP is preferred in a slow LAN, WAN, or dial-up network, whereas named pipes can be a better choice when network speed is not the issue, as it offers more functionality, ease of use, and configuration options.

Did u know? By default, the only network protocols enabled ob most of the edition of SQL server are TCP/IP and Shared memory.

Self Assessment

Fill in the blanks:

1. ..... is the simplest protocol to use and has no configurable settings.
2. ..... includes standards for routing network traffic and offers advanced security features.
3. Named Pipes is a protocol developed for .....
4. VIA stands for ..... protocol
5. By default, the only network protocols enabled on most of the edition of SQL server are .....
6. Clients that use MDAC 2.8 or earlier cannot use ..... protocol.

8.2 Enabling and Configuring Protocols

SQL Server Configuration Manager incorporates the functions of Client Network Utility from earlier versions of SQL Server and provides client-configuration functionality for any version of SQL Server compatible with SQL Server.

To launch SQL Server Configuration Manager, click Start, point to Program Files, point to Microsoft SQL Server, point to Configuration Tools, and then click SQL Server Configuration Manager. To access the client configuration section, in the console pane click SQL Server Native Client Configuration.

Notes

Use SQL Server Configuration Manager to:

- Change the default protocol, used by clients when attempting to connect to a server.
- Change the order in which all enabled protocols are attempted, when connecting to a server.
- Create client connections to specified servers and save them as configuration entries. Configuration entries consist of a server alias, a client protocol, and any relevant connection parameters, such as a pipe name, or port number.
- Display information about the SQL Server client protocols currently installed on the system.

Notes SQL Server Configuration Manager creates registry entries for the server alias configurations and default client network library. However, the application does not install either the SQL Server client network libraries or the network protocols. The SQL Server client network libraries are installed during SQL Server Setup; the network protocols are installed as part of Microsoft Windows Setup (or through Networks in Control Panel). A particular network protocol may not be available as part of Windows Setup. For more information about installing these network protocols, see the vendor documentation.

Task List the steps to launch SQL Server Configuration Manager

8.3 Native Client Configuration

SQL Server Native Client is the network library that client computers use to connect to SQL Server, starting with Microsoft SQL Server.

The settings configured in SQL Server Native Client Configuration, are used on the computer running the client program. When configured on the computer running SQL Server, they affect only those client programs running on the server.

These settings do not affect clients connecting to previous versions of SQL Server, unless they are using the client tools starting with SQL Server, such as SQL Server Management Studio

8.3.1 Client Protocols Properties (Order Tab)

Use the Orderpage on the Client Protocols Properties dialog box to view and enable the client protocols.

Click a protocol, and then click Enable or Disable to move the selected protocol to the Disabled Protocols or Enabled Protocols list.

Protocols are tried in the order listed, attempting to connect using the top protocol first, and then the second listed protocol, etc. Move protocols up or down the Enabled Protocols list, by clicking the up arrow and down arrow buttons. When connecting to Microsoft SQL Server from a client on that computer, the Shared Memory protocol will always be tried first, if enabled.

Notes These settings are not used by Microsoft .NET SqlClient. The protocol order for .NET SqlClient is first TCP, and then named pipes, which cannot be changed.

Notes

Options

1. ***Disabled Protocols:*** Lists protocols which are installed but are not currently used.
2. ***Enabled Protocols:*** Lists protocols which are available for Microsoft SQL Server clients on this computer.
   i. > Enables the currently highlighted protocol in the Disabled Protocols box, moving it to the Enabled Protocols box.
   ii. < Disables the currently highlighted protocol in the Enabled Protocols box, moving it to the Disabled Protocols box.

   Up arrow: Moves the highlighted protocol up in the list. This allows you to increase the priority in which the Net-Library will attempt to use the selected protocol for connections.

   Down arrow: Moves the highlighted protocol down in the list. This allows you to decrease the priority in which the Net-Library will attempt to use the selected protocol for connections.
3. ***Enable Shared Memory Protocol:*** Enables the shared memory protocol which is always tried first (if enabled), when connecting to SQL Server from a client on that computer.

Notes If the protocol is specified through a prefix or as part of the connection string, only the specified protocol is attempted..

8.3.2 Client Protocols – Shared Memory Properties (Protocol Tab)

In Microsoft SQL Server Configuration Manager use the Protocol tab on the Shared Memory Properties dialog box to view or modify shared memory.

8.3.3 Client Protocols – TCP/IP Properties (Protocol Tab)

In Microsoft SQL Server Configuration Manager, use the Protocol tab on the TCP/IP Properties dialog box to view or specify the following options. To connect to a different port, type the port number in the Default Port box.

Default Port: Specifies the port that the TCP/IP Net-library will use to attempt to connect to the target instance of SQL Server. The default value port is 1433.

When connecting to a default instance of Database Engine, the client uses this value. If a default instance has been configured to listen on a different port, change this value to that port number.

When connecting to a named instance of Database Engine, the client will attempt to obtain the port number from the SQL Server Browser Service running on the server computer. If the SQL Server Browser Service is not running, the port number must be provided through this setting, or as part of the connection string.

Enabled: Possible values are Yes and No.

Keep Alive: This parameter (in milliseconds) controls how often TCP attempts to verify that an idle connection is still intact by sending a KEEPALIVE packet. The default is 30000 milliseconds.

Keep Alive Interval: This parameter (in milliseconds) determines the interval separating KEEPALIVE retransmissions until a response is received. The default is 1000 milliseconds

Notes

8.3.4 Client Protocols – Named Pipes Properties (Protocol Tab)

In Microsoft SQL Server Configuration Manager use the Protocol tab on the Named Pipes Properties dialog box to view or modify the description of default pipe. To connect to a different pipe, type the pipe in the Default Pipe box.

Options

Default Pipe: Specifies the default pipe the Named Pipes Net-library will use to attempt to connect to the target instance of SQL Server. By default, SQL Server listens on: \\.\pipe\sql\query

To connect to the default pipe, enter sql\query

Enabled: Possible values are Yes and No.

8.3.5 Client Protocols – VIA Properties (Protocol Tab)

In Microsoft SQL Server Configuration Manager, use the Protocol tab on the VIA Properties dialog box to view or specify the following options for the Virtual Interface Adapter.

Options

Default NIC: Indicates which Network Interface Card (NIC) the VIA protocol is bound to. NICs are numbered, starting at zero. Computers with only one NIC will indicate 0.

Default Server: VIA port that VIA is listening on when accepting connections from VIA clients.

Enabled: Possible values are Yes and No.

Self Assessment

State true or false:

7. To connect to the default pipe, enter sql\query
8. NIC stands for Network Indicator Card.
9. SQL Server Native Client is the network library that client computers use to connect to SQL Server, starting with Microsoft SQL Server.
10. The settings configured in SQL Server Native Client Configuration, are used on the computer running the server program.
11. The default value of KEEP ALIVE parameter is 40000 milliseconds.

8.4 Summary

- Server network configuration tasks include enabling protocols, modifying the port or pipe used by a protocol, configuring encryption, configuring the SQL Server Browser service, exposing or hiding the Microsoft Database Engine on the network, and registering the Server Principal Name.
- Shared memory is the simplest protocol to use and has no configurable settings. Clients that use MDAC 2.8 or earlier cannot use this protocol.
- TCP/IP includes standards for routing network traffic and offers advanced security features.
- Named Pipes is a protocol developed for local area networks. For named pipes, network communications are typically more interactive

Notes

- VIA stands for Virtual Interface Adapter (VIA) protocol.
- By default, the only network protocols enabled ob most of the edition of SQL server are TCP/IP and Shared memory.
- SQL Server Native Client is the network library that client computers use to connect to SQL Server, starting with Microsoft SQL Server.

8.5 Keywords

Keep Alive: This parameter (in milliseconds) controls how often TCP attempts to verify that an idle connection is still intact by sending a KEEPALIVE packet.

Keep Alive Interval: This parameter (in milliseconds) determines the interval separating KEEPALIVE retransmissions until a response is received.

Named Pipes: Named Pipes is a protocol developed for local area networks.

SQL Server Native Client: SQL Server Native Client is the network library that client computers use to connect to SQL Server, starting with Microsoft SQL Server.

TCP/IP: TCP/IP includes standards for routing network traffic and offers advanced security features.

VIA: VIA stands for Virtual Interface Adapter (VIA) protocol.

8.6 Review Questions

1. Explain the different types of network protocols used in SQL server.
2. Describe the process to configure and secure the network protocols.
3. Explain the use and functions of SQL Configuration Manager in network configuration.
4. Discuss the process to configure native client configuration in SQL Server.
5. Explain the functions of SQL Server Configuration Manager.
6. Differentiate between named pipes and TCP/IP sockets.
7. Illustrate the working of Virtual Interface Adapter (VIA).
8. Explain the concept of enabling and configuring protocols.

Answers: Self Assessment

1. Shared memory
2. TCP/IP
3. local area networks
4. Virtual Interface Adapter
5. TCP/IP and Shared memory.
6. shared memory
7. True
8. False
9. False
10. True
11. False

8.7 Further Readings

Notes

Books

C.J. Date, *Introduction to Database Systems*, Pearson Education.

Elmasri Navrate, *Fundamentals of Database Systems*, Pearson Education.

Peter Rob & Carlos Coronel, *Database Systems Design, Implementation and Management*, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, *Database Management Systems*, Tata McGraw Hill, 3rd Edition.

Silberschatz, Korth, *Database System Concepts*, McGraw Hill, 5th Edition.

Online links

www.sql-tutorial.net/

www.w3schools.com/sql/

www.sqlservertutorials.com/

Notes

Unit 9: Database Recovery Models

CONTENTS

Objectives
Introduction
9.1 Common Terms
9.2 Recovery Models
9.3 Partial Database Restore
9.4 How to View or Change the Recovery Model of a Database
9.4.1 To View or Change the Recovery Model of a Database
9.4.2 Considerations for Switching from the Simple Recovery Model
9.4.3 Considerations for Switching to the Full Recovery Model
9.4.4 Considerations for Switching to the Simple Recovery Model
9.4.5 Switching between Full and Bulk-Logged Recovery
9.4.6 Switching from Full or Bulk-Logged to Simple Recovery
9.5 Changing the Recovery Model
9.6 Summary
9.7 Keywords
9.8 Review Questions
9.9 Further Readings

Objectives

After studying this unit, you will be able to:

- Know about the common terms in database backup
- Discuss the various recovery models in SQL server
- Learn how to View or Change the Recovery Model of a Database
- Understand the process of switching from Full or Bulk-Logged to Simple Recovery

Introduction

The need to back up databases on a regular basis is a major component of managing any production system. Backups may be used to provide a means of recovery from a disaster situation. Microsoft® SQL Server™ 2000 provides several kinds of recovery models that may be combined to formulate a customized disaster recovery plan depending on the nature of the data and the disaster recovery requirements. SQL Server 2000 enhances some aspects of the backup and restore functionality that was provided in SQL Server 7. There is also additional functionality that helps individual organizations take full advantage of commands in SQL Server 2000.

9.1 Common Terms

1. ***Data page:*** An SQL Server database's basic data storage structure is 8 KB and is known as a *data page*. An SQL database may contain thousands of pages.

Notes

2. ***Disaster recovery planning:*** The process of formulating, documenting, and testing the procedures that would be performed if production data in one or more SQL Server databases were to be lost or modified in an unforeseen disaster or malicious attack.
3. ***Minimally logged operations (bulk load operations):*** Data movement operations that require minimal logging in the transaction log. These operations include **bcp**, certain Data Transformation Services (DTS) operations, and SELECT INTO. Depending on the recovery model for a database, any of these operations might either be fully logged or minimally logged.
4. ***Filegroup:*** A logical grouping of SQL Server database files. By default, a new SQL Server database contains the Primary filegroup.
5. ***Log Sequence Number (LSN):*** The unique number that each operation is stamped with when it is written to the transaction log. A single SQL Server transaction may contain several LSNs.
6. ***Logical file names:*** The names that are used by SQL Server to identify files within an SQL Server database.
7. ***Physical file names:*** The name used by the operating system to identify specific files. All SQL Server database files have both a physical and a logical file name.
8. ***Extent:*** A collection of 8 data pages. Since a data page is 8 KB, an extent is 64 KB.
9. ***SQL Query Analyzer:*** A graphical tool provided with SQL Server client utilities to query SQL Server databases using the Transact-SQL commands.
10. ***Transaction:*** A set of modifications that are performed as a single unit of work. A transaction follows the ACID guidelines. For more information on the ACID standard, see "Transactions" in SQL Server Books Online.
11. ***Transaction log:*** A record of modifications performed to a database. The amount of information logged in the transaction log depends on the recovery model for a database.
12. ***Tail of transaction log:*** The transactions that have been committed but not backed up since the previous complete or differential database backup or transaction log backup.
13. ***Transaction undo file:*** File containing information regarding any modifications that were made as part of incomplete transactions at the time the backup was performed. A transaction undo file is required if a database is loaded in read-only state. In this state, further transaction log backups may be applied.
14. ***Virtual Log File (VLF):*** A logical section within an SQL Server database's transaction log. When performing a truncate of the transaction log, an entire VLF is cleaned out.

Self Assessment

Name the following:

1. A logical section within an SQL Server database's transaction log.
2. A set of modifications that are performed as a single unit of work.
3. The unique number that each operation is stamped with when it is written to the transaction log.
4. A graphical tool provided with SQL Server client utilities to query SQL Server databases using the Transact-SQL commands.

Notes

9.2 Recovery Models

SQL Server 2000 introduces the concept of recovery models for databases. Recovery models are designed to simplify the administration of SQL Server 2000 databases. There are three recovery models in SQL Server 2000—Full, Bulk-Logged, and Simple. System databases (including **master**, **MSDB**, and **tempdb**) are set to the Simple Recovery model. All user databases, by default, are created with the Full Recovery model (it should be noted that the Full Recovery model takes affect once a complete database backup is performed). The recovery model may be changed once the database is created.

The recovery model for a database incorporates the two most often used settings—Truncate Log on Checkpoint and Select Into/Bulk copy.

1. ***Truncate Log on Checkpoint***. In previous versions of SQL Server, this setting was selected to automatically truncate the transaction log every time CHECKPOINT is activated for the database.
2. ***Select Into/Bulkcopy***. This setting was used in previous versions of SQL Server to perform non-logged operations.

Following are the settings and their relation to the three recovery models.

Recovery Model	Select Into / BulkCopy	Truncate Log on Checkpoint
Full	False	False
Bulk-Logged	True	False
Simple	True/False	True

1. ***Simple Recovery Model:*** The simple recovery model minimizes administrative overhead for the transaction log, because the transaction log is not backed up. The simple recovery model risks significant work-loss exposure if the database is damaged. Data is recoverable only to the most recent backup of the lost data. Therefore, under the simple recovery model, the backup intervals should be short enough to prevent the loss of significant amounts of data. However, the intervals should be long enough to keep the backup overhead from affecting production work. Including differential backups in the backup strategy can help reduce the overhead.

   Generally, for a user database, the simple recovery model is useful for test and development databases or for databases containing mostly read-only data, such as a data warehouse. The simple recovery model is inappropriate for production systems where loss of recent changes is unacceptable. In such cases, we recommend using the full recovery model.

2. ***Bulk-Logged Recovery Model:*** The full recovery and bulk-logged recovery models provide greater protection for data than the simple recovery model. These recovery models rely on backing up the transaction log to provide full recoverability and to prevent work loss in the broadest range of failure scenario.

   This recovery model bulk logs most bulk operations. It is intended solely as an adjunct to the full recovery model. For certain large-scale bulk operations such as bulk import or index creation, switching temporarily to the bulk-logged recovery model increases performance and reduces log space consumption. Log backups are still required. Like the full recovery model, the bulk-logged recovery model retains transaction log records until after they are backed up. The tradeoffs are bigger log backups and increased work-loss exposure because the bulk-logged recovery model does not support point-in-time recovery.

3. ***Full Recovery Model:*** SQL Server performs full transaction logging for any bulk load operations if a database is in Full Recovery model. Transaction log backups should be performed at regular intervals for maximum recoverability. This model provides the safest mode of operation for production systems.

Notes

Provides the normal database maintenance model for databases where durability of transactions is necessary.

Log backups are required. This model fully logs all transactions and retains the transaction log records until after they are backed up. The full recovery model allows a database to be recovered to the point of failure, assuming that the tail of the log can be backed up after the failure. The full recovery model also supports restoring individual data pages.

Notes Under the full recovery and bulk-logged recovery models, log backups are essential. If you do not want to take log backups, use the simple recovery model.

Difference between various recovery models

The following table summarizes the recovery models and backup types available with each recovery model.

Recovery Model/ Backup	Complete	Differential	Transaction Log	File / Filegroup
Simple	Required	Allowed	Not Allowed	Not Allowed
Bulk-Logged	Required	Allowed	Required	Allowed
Full	Required	Allowed	Required	Allowed

9.3 Partial Database Restore

New functionality in SQL Server 2000 provides commands to restore a database backup partially. If a database contains several filegroups, a single filegroup may be recovered using this new functionality. Partial database restore operations provide a means to restore only certain parts of the database, as needed.

Self Assessment

Fill in the blanks:

5. An SQL Server database's basic data storage structure is 8 KB and is known as a …….
6. ……. should be performed at regular intervals for maximum recoverability.
7. There are three recovery models in SQL Server 2000—……., Bulk-Logged and Simple.
8. A database in ……. recovery model will have minimum logging for bulk import operations.
9. Recovery models are designed to simplify the ……. of SQL Server 2000 databases.

9.4 How to View or Change the Recovery Model of a Database

Caution Before changing the recovery model of a database, you should understand the impact of the change on your backup and restore strategy.

Notes

9.4.1 To View or Change the Recovery Model of a Database

1. After connecting to the appropriate instance of the Microsoft SQL Server Database Engine, in Object Explorer, click the server name to expand the server tree.
2. Expand Databases, and, depending on the database, either select a user database or expand System Databases and select a system database.
3. Right-click the database, and then click Properties, which opens the Database Properties dialog box.
4. In the Select a Page pane, click Options.
5. The current recovery model is displayed in the Recovery model list box.
6. Optionally, to change the recovery model select a different model list. The choices are Full, Bulk-logged, or Simple.

9.4.2 Considerations for Switching from the Simple Recovery Model

A database can be switched to another recovery model at any time. However, switching from the simple recovery model is unusual. Be aware that if you switch to the full recovery model during a bulk operation, the logging of the bulk operation changes from minimal logging to full logging, and vice versa.

9.4.3 Considerations for Switching to the Full Recovery Model

If you must switch from the simple recovery model to the full recovery model, we recommend that you:

1. Immediately after you complete the switch to the full recovery model or bulk-logged recovery model, take a full or differential database backup to start the log chain.

   The switch to the full or bulk-logged recovery model takes effect only after the first data backup.
2. Schedule regular log backups and update your restore plan accordingly.

   Log backups are an integral and fundamental aspect of the full and bulk-logged recovery models. Log backups allow the transaction log to be truncated. If you do not back up the log frequently enough, the transaction log can expand until it runs out of disk space.

9.4.4 Considerations for Switching to the Simple Recovery Model

If you switch from the full or bulk-logged recovery model to the simple recovery model, you break the backup log chain. Therefore, we strongly recommend that you back up the log immediately before switching, which allows you to recover the database up to that point. After switching, you need to take periodic data backups to protect your data and to truncate the inactive portion of the transaction log.

9.4.5 Switching between Full and Bulk-Logged Recovery

For a database that uses full recovery, switching to the bulk-logged recovery model temporarily for bulk operations improves performance. However, if data loss is unacceptable, to prevent data loss, we recommend that you switch to the bulk-logged recovery model only under the following conditions:

Notes

- Users are currently not allowed in the database.
- No modifications are made during bulk processing that are not recoverable without depending on taking a log backup; for example, by re-running the bulk processes.

We recommend that:

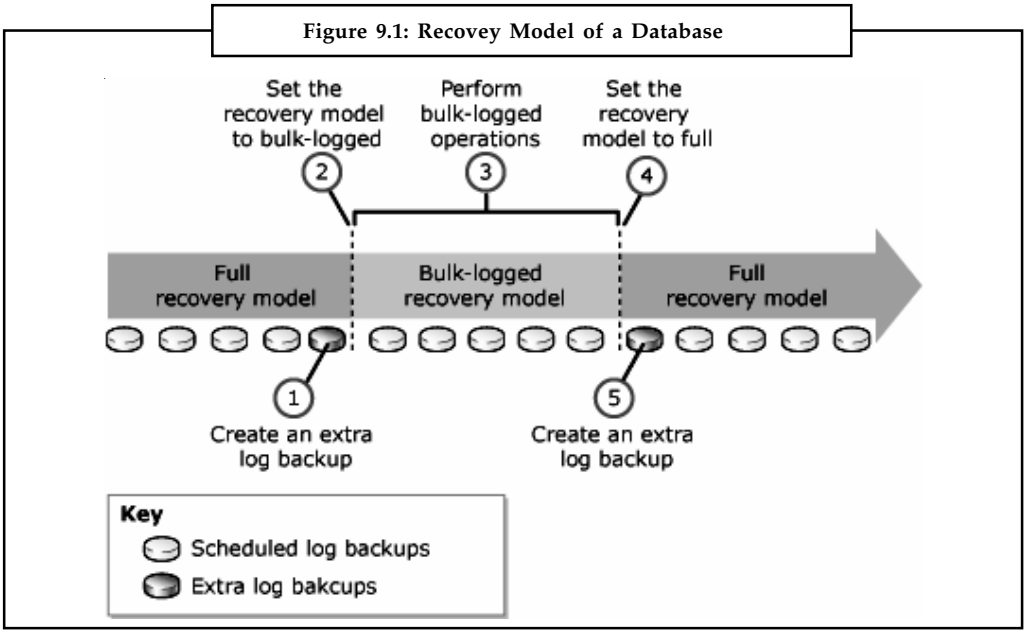
- Before switching to the bulk-logged recovery model, you back up the log.

This is important because, under the bulk-logged recovery model, if the database fails, backing up the log for bulk operations requires access to the data.

- After performing the bulk operations, you immediately switch back to full recovery mode.
- After switching back from the bulk-logged recovery model to the full recovery model, you back up the log again.

Following these recommendations fully protects your data and enables point-in-time recovery. The following figure illustrates these recommendations.

Figure 9.1: Recovey Model of a Database



When switching between two recovery models, your backup strategy remains the same: continue performing periodic database, log, and differential backups.

9.4.6 Switching from Full or Bulk-Logged to Simple Recovery

Switching from the full or bulk-logged recovery to simple recovery is possible, but uncommon.

Back up the transaction log just before switching to the simple recovery model, to permit recovery to that point. Backing up the log is not supported under the simple recovery model, so, after switching, discontinue any scheduled jobs for backing up the transaction log.

Task Describe the procedure of **Switching Between Full and Bulk-Logged Recovery**

Notes

9.5 Changing the Recovery Model

To change the recovery model (Transact-SQL)

Use ALTER DATABASE, as follows:

- To set the database to the full recovery model:

USE master;

ALTER DATABASE database_name SET RECOVERY FULL;

- To set the database to the bulk-logged recovery model:

USE **master**;

ALTER DATABASE database_name SET RECOVERY BULK_LOGGED;

Notes To change the default recovery model for new databases, use ALTER DATABASE to change the recovery model of the model database.

9.6 Summary

- Simple recovery model facilitates the maintenance of a database by making the transaction log virtually maintenance free. There are limitations placed on the recoverability of a database if this recovery model is used.
- A database in this recovery model will have minimum logging for bulk import operations. Space allocation and deallocation is only logged for bulk import operations. Point-in-time and point-of-failure recovery may be possible when a database is in Bulk-Logged Recovery model.
- The full recovery and bulk-logged recovery models provide greater protection for data than the simple recovery model. These recovery models rely on backing up the transaction log to provide full recoverability and to prevent work loss in the broadest range of failure scenario.
- A database can be switched to another recovery model at any time. However, switching from the simple recovery model is unusual. Be aware that if you switch to the full recovery model during a bulk operation, the logging of the bulk operation changes from minimal logging to full logging, and vice versa.

9.7 Keywords

Data Page: An SQL Server database's basic data storage structure is 8 KB and is known as a *data page*

***Disaster Recovery Planning*:** The process of formulating, documenting, and testing the procedures that would be performed if production data in one or more SQL Server databases were to be lost or modified in an unforeseen disaster or malicious attack.

Extent: A collection of 8 data pages. Since a data page is 8 KB, an extent is 64 KB.

File Group: A logical grouping of SQL Server database files. By default, a new SQL Server database contains the Primary file group.

Notes

Log Sequence Number (LSN): The unique number that each operation is stamped with when it is written to the transaction log. A single SQL Server transaction may contain several LSNs.

Simple Recovery Model: The simple recovery model minimizes administrative overhead for the transaction log, because the transaction log is not backed up.

SQL Query Analyzer: A graphical tool provided with SQL Server client utilities to query SQL Server databases using the Transact-SQL commands.

Transaction: A set of modifications that are performed as a single unit of work. A transaction follows the ACID guidelines

Transaction Log: A record of modifications performed to a database. The amount of information logged in the transaction log depends on the recovery model for a database.

Tail of Transaction Log: The transactions that have been committed but not backed up since the previous complete or differential database backup or transaction log backup.

Transaction undo File: File containing information regarding any modifications that were made as part of incomplete transactions at the time the backup was performed. A transaction undo file is required if a database is loaded in read-only state. In this state, further transaction log backups may be applied.

Virtual Log File (VLF)**:** A logical section within an SQL Server database's transaction log. When performing a truncate of the transaction log, an entire VLF is cleaned out.

9.8 Review Questions

1. Explain the need of database recovery.
2. Describe the various recovery models in SQL server.
3. Discuss Simple recovery model.
4. How will you use the full recovery model for database recovery?
5. Differentiate between the three database recovery model in SQL server.
6. Describe the process of changing the Recovery Model.
7. Explain the process of Switching from Full or Bulk-Logged to Simple Recovery.
8. Analyze the various terms used for database recovery model.
9. What are the two frequently used settings that database incorporates in the recovery model? Discuss.
10. Illustrate the steps required in viewing or changing the recovery model of a database.

Answers: Self Assessment

1. Virtual Log File
2. Transactions
3. Log Sequence Number
4. SQL Query Analyzer
5. data page
6. Transaction log backups
7. Full
8. Bulk-Logged
9. Administration

Notes

9.9 Further Readings

Books

C.J. Date, *Introduction to Database Systems*, Pearson Education.

Elmasri Navrate, *Fundamentals of Database Systems*, Pearson Education.

Peter Rob & Carlos Coronel, *Database Systems Design, Implementation and Management*, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, *Database Management Systems*, Tata McGraw Hill, 3rd Edition.

Silberschatz, Korth, *Database System Concepts*, McGraw Hill, 5th Edition.

Online links

www.sql-tutorial.net/

www.w3schools.com/sql/

www.sqlservertutorials.com/

Notes

Unit 10: Database Backup and Restore

CONTENTS

Objectives
Introduction
10.1 Backup Types and Options
10.1.1 Types of Backups
10.1.2 Backup Media
10.1.3 Permissions Required for Backup and Restore
10.2 Complete Database Backup
10.2.1 Performing Complete Database Backups
10.3 Restoring Database
10.3.1 Restoring a Complete Backup to the Same Database
10.4 Database Snapshots
10.4.1 Meaning of Database Snapshot
10.4.2 How Database Snapshots Work
10.4.3 Effect of the Update Pattern on Database Snapshot Growth
10.4.4 Metadata about Database Snapshots
10.4.5 Typical Uses of Database Snapshots
10.4.6 Reasons to take Database Snapshots
10.4.7 Limitations and Requirements of Database Snapshots
10.4.8 Creating a Database Snapshot
10.5 Summary
10.6 Keywords
10.7 Review Questions
10.8 Further Readings

Objectives

After studying this unit, you will be able to:

- Learn about the basic data backup models
- Know how backup functionality can be used in a disaster recovery plan
- Describe various types of backup media
- Understand the concept of database snapshot and its use
- Learn how to create a database snapshot

Notes

Introduction

In the last unit you have studied about database recovery models. In this unit, you will learn about back up types and options. This unit will help you understand the various types of backup strategies available in SQL Server enterprise.

It is highly recommended that all SQL Server databases be backed up periodically. This provides the best chance of successfully recovering a production environment in the quickest amount of time in case there is a disaster situation.

This unit discusses the various kinds of backups that are available in SQL Server 2000 and how this functionality may be used in a disaster recovery plan. The unit also discusses some general recommendations on how to improve backup and restore throughput.

10.1 Backup Types and Options

A copy of data that can be used to restore and recover the data is called a backup. Backups let you restore data after a failure. With good backups, you can recover from many failures, such as:

- Media failure.
- User errors, for example, dropping a table by mistake.
- Hardware failures, for example, a damaged disk drive or permanent loss of a server.
- Natural disasters.

Additionally, backups of a database are useful for routine administrative purposes, such as copying a database from one server to another, setting up database mirroring, and archiving.

10.1.1 Types of Backups

SQL Server provides several different kinds of backups. A combination of these backups may be used to formulate a robust disaster recovery strategy.

Scope of Backup

The scope of a backup of data (a data backup) can be a whole database, a partial database, or a set of files or filegroups. For each of these, SQL Server supports full and differential backups:

1. ***Full backup:*** A full backup contains all the data in a specific database or set of filegroups or files, and also enough log to allow for recovering that data.
2. ***Differential backup:*** A differential backup is based on the latest full backup of the data. This is known as the base of the differential, or the differential base. A differential base is a full backup of read/write data. A differential backup contains only the data that has changed since the differential base. Typically, differential backups that are taken fairly soon after the base backup are smaller and faster to create than the base of a full backup. Therefore, using differential backups can speed up the process of making frequent backups to decrease the risk of data loss. Usually, a differential base is used by several successive differential backups. At restore time, the full backup is restored first, followed by the most recent differential backup.

Over time, as a database is updated, the amount of data that is included in differential backups increases. This makes the backup slower to create and to restore. Eventually, another full backup will have to be created to provide a new differential base for another series of differential backups.

Notes

Notes Typically, a differential backup covers the same data files as those files that are covered in a single differential base. Under the simple recovery model, a differential backup can have only one differential base. Trying to use multiple bases causes an error and the backup operation fails. Under the full recovery model, differential file backups can use multiple bases, but this can be difficult to manage.

Each data backup includes part of the transaction log so that the backup can be recovered to the end of that backup.

After the first data backup, under the full recovery model or bulk-logged recovery model, regular transaction log backups (or log backups) are required. Each log backup covers the part of the transaction log that was active when the backup was created, and the log backup includes all log records that were not backed up in a previous log backup.

Database Backups

Database backups are easy to use and are recommended whenever database size allows. SQL Server supports the following types of database backups.

Backup type	Description
Database backup	A full backup of the whole database. Database backups represent the whole database at the time the backup finished.
Differential database backups	A backup of all files in the database. This backup contains only the data extents that were modified since the most recent database backup of each file.

Partial Backups

Partial and differential partial backups were introduced in SQL Server 2005. These backups are designed to provide more flexibility for backing up databases that contain some read-only filegroups under the simple recovery model. However, these backups are supported by all recovery models.

SQL Server 2008 supports the following types of file backups.

Backup type	Description
Partial backup	A backup of all the full data in the primary filegroup, every read/write filegroup, and any optionally specified read-only files or filegroups. A partial backup of a read-only database contains only the primary filegroup.
Differential partial backup	A backup that contains only the data extents that were modified since the most recent partial backup of the same set of filegroups.

File Backups

The files in a database can be backed up and restored individually. Using file backups can increase the speed of recovery by letting you restore only damaged files, without restoring the rest of the database. For example, if a database consists of several files that are located on different disks and one disk fails, only the file on the failed disk has to be restored. However, planning and restoring file backups can be complex; therefore, file backups should be used only where they clearly add value to your restore plan.

Notes

SQL Server supports the following types of file backups.

Backup type	Description
File backup	A full backup of all the data in one or more files, or filegroups. **Important** Under the simple recovery model, file backups are basically restricted to read-only secondary filegroups. You can create a file backup of a read/write filegroup, but before you can restore the read/write file backup, you must set the filegroup to read-only and take a differential read-only file backup.
Differential file backups	A backup of one or more files that contain data extents that were changed since the most recent full backup of each file. **Note** Under the simple recovery model, this assumes that the data has been changed to read-only since the full backup.

Under the full recovery model or bulk-logged recovery model, regular transaction log backups (or log backups) are required. Each log backup covers the part of the transaction log that was active when the backup was created, and it includes all log records that were not backed up in a previous log backup. An uninterrupted sequence of log backups contains the complete log chain of the database, which is said to be unbroken. Under the full recovery model, and sometimes under the bulk-logged recovery model, an unbroken log chain lets you to restore the database to any point in time.

Before you can create the first log backup, you must create a full backup, such as a database backup. Thereafter, backing up the transaction log regularly is necessary, not only to minimize work-loss exposure but also to enable truncation of the transaction log.

Notes To limit the number of log backups that you need to restore, it is essential to routinely back up your data. For example, you might schedule a weekly full database backup and daily differential database backups.

Copy-Only Backups

Usually, taking a backup changes the database and affects how later backups are restored. However, occasionally, it is useful to take a backup for a special purpose without affecting the overall backup and restore procedures for the database. For this purpose, copy-only backups were introduced in SQL Server 2005. These backups are independent of the regular sequence of SQL Server backups.

Summary of back up types in SQL server

Backup Type	Description
Complete	Backs up the entire database.
Differential	Backs up only modified extents since the previous complete backup.
Transaction Log	Backs up the active portion and truncates the inactive portion of the transaction log.
File / Filegroup	Backs up individual files and filegroups within a database.
File differential	Combines differential backups and file or filegroup backups.

Self Assessment

Notes

Name the following:

1. A backup contains all the data in a specific database or set of filegroups or files, and also enough log to allow for recovering that data.
2. A backup that is based on the latest full backup of the data and is a full backup of read/write data.
3. A backup of all the full data in the primary filegroup, every read/write filegroup, and any optionally specified read-only files or filegroups.
4. A backup that contains only the data extents that were modified since the most recent partial backup of the same set of filegroups.
5. A full backup of all the data in one or more files, or filegroups.

10.1.2 Backup Media

SQL Server databases may be backed up to either a disk or tape media. Backup may be performed through SQL Server Enterprise Manager or a Transact-SQL command.

1. ***Disk Backups:*** A database may be backed up to disk file or a disk backup device. Any database can be backed up to a random disk file at any time. The file may either be initialized or the backup may be appended to an existing backup file.
2. ***Tape Backups:*** A database may be backed up to a local tape drive. SQL Server formats the tape backups using Microsoft Tape Format (MTF). This means that a tape may hold other backups formatted using MTF in conjunction with SQL Server backups. Tape backups provide certain features that are not available when using disk backups.
3. ***Continuation Media:*** If the tape to which the backup is being written fills up, SQL Server Enterprise Manager pops a dialog box and prompts for the next tape (if using the Transact-SQL command, a message is logged to the SQL Server error log to mount the next tape and a retry attempt is made roughly every five minutes to see if a new tape was mounted). This is in contrast to disk backups, where inadequate disk space terminates the backup operation.
4. ***Restart Option:*** If there is a power failure or the server shuts down unexpectedly while the backup/restore is being performed, the operation may be restarted from the point at which it was interrupted.

10.1.3 Permissions Required for Backup and Restore

Any logon that requires permissions to perform backup or restore operations should be provided membership in the following SQL Server roles:

Server Role : sysadmin

DB role : db_backupoperator, dbo_owner

Permissions required for performing restore -

Server role : sysadmin, dbcreator

DB role : db_owner

Notes

10.2 Complete Database Backup

A complete database backup creates a stand-alone image of the entire database. A complete database backup is self-dependent and may be restored to either the same or a new database on the same or a different server. This provides plenty of flexibility at the time when this backup has to be restored.

A complete backup may be restored without the need for any other kind of backup. It may also be performed for databases in any recovery model. Restoring a complete database backup typically would be considered a starting point for a disaster recovery situation where the entire database is lost or damaged.

It is recommended that a complete database backup be performed at regular intervals for all production databases. It is also recommended that a complete backup should be performed for system databases (including **master** and **MSDB**) if there are any changes performed to the SQL Server operating environment such as creating or removing databases, configuring security, creating and modifying DTS packages or scheduled jobs, adding and removing linked servers, etc.

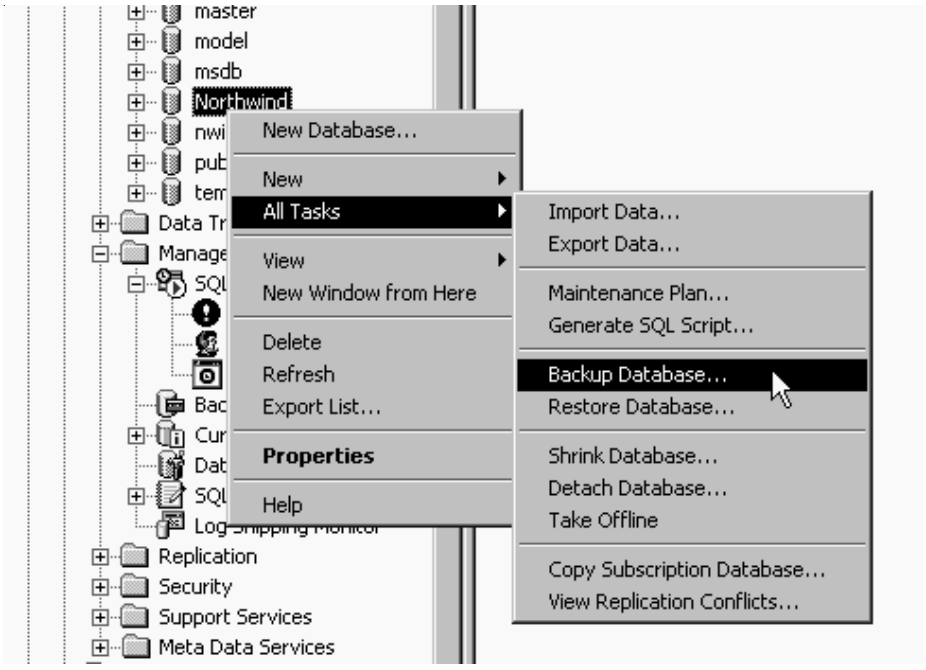
10.2.1 Performing Complete Database Backups

A complete database backup may be performed either through SQL Server Enterprise Manager or by using Transact-SQL commands. Complete backups may also be scheduled to be performed at regular intervals. Scheduling may be done through either SQL Server Enterprise Manager or using Transact-SQL commands.

To perform a complete database backup through SQL Server Enterprise Manager

1. Open Enterprise Manager and connect to the server.
2. Expand the **Databases** folder, then right-click the database that you want to back up.

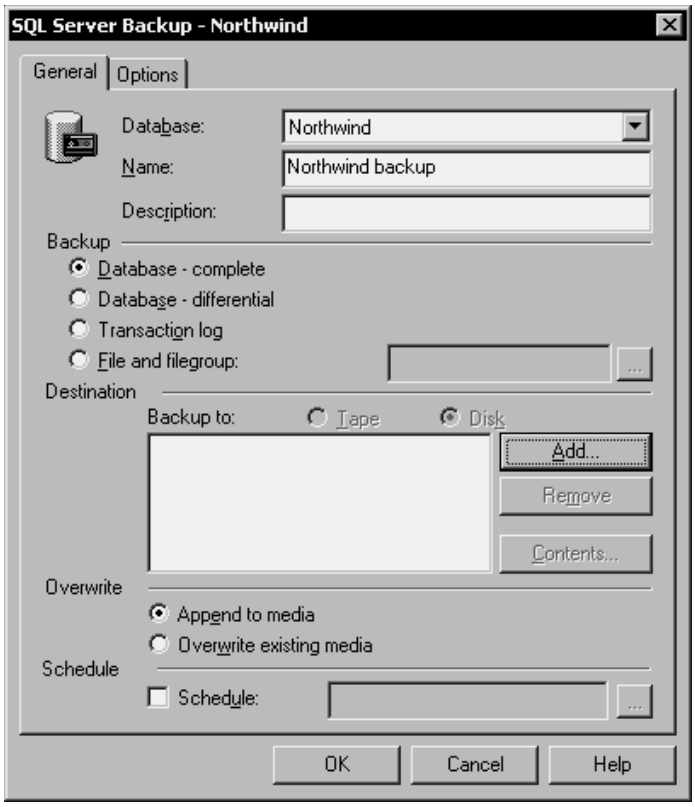
Figure 10.1: Backup Database through Enterprise Manager



Notes

3. Select **All Tasks**, then select **Backup Database**... as shown in Figure 10.2. The dialog box shown in Figure 10.2 is displayed.

Figure 10.2: Backup Database Dialog Box

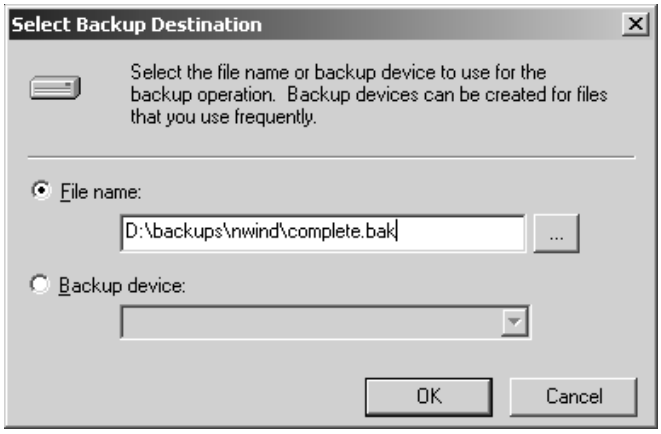


4. Provide a name for the backup in the **Name** text box. Leave the **Database – complete** radio button selected since we are performing a complete database backup.

5. Select the **Overwrite existing media** check box to initialize the destination file or device or select the **Append to media** check box to append the current backup to existing file or device.

6. To select a destination for the backup, click the **Add** button. The dialog box shown in Figure 10.3 is displayed.

Figure 10.3: Select Backup Destination

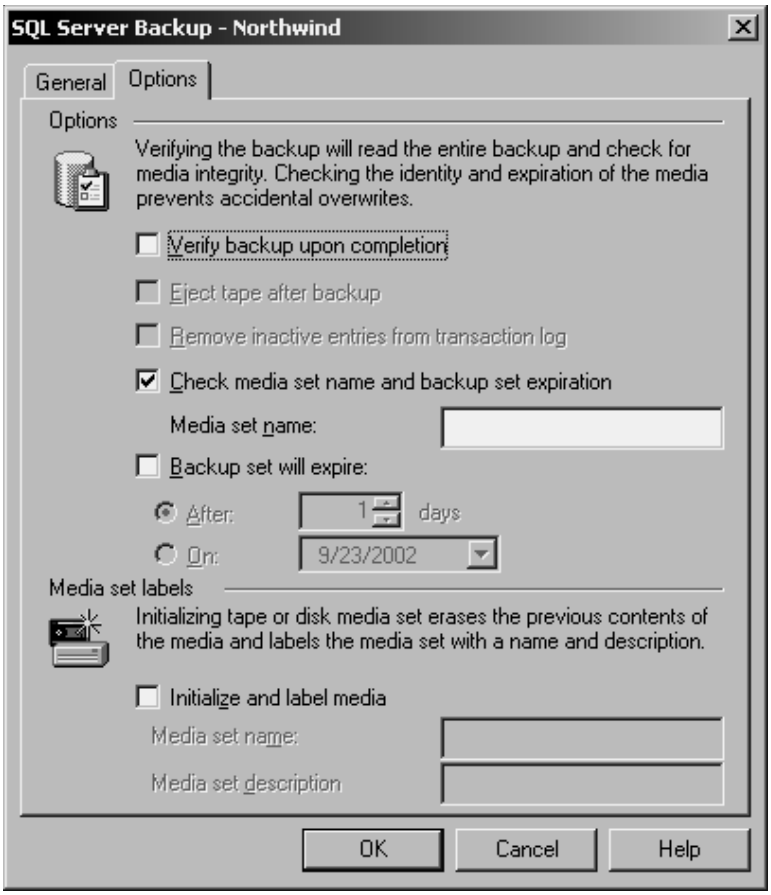


Notes

7. Select an existing file or enter a new file name. Click **OK** after selecting a file.

8. Click the **Options** tab. The options shown in Figure 10.4 are presented.

Figure 10.4: SQL Server Backup Options



9. Select the **Verify backup upon completion** check box to verify the backup upon completion.

Description of other fields

Remove inactive entries from transaction log: truncates the transaction log while performing the backup. If this setting is not checked, SQL Server uses the NO_TRUNCATE option for the backup. This option is available only while performing transaction log backup.

Check media set name and backup set expiration: verifies the selected media for the provided media set name to prevent accidental overwrites.

Eject tape after backup: ejects the tape from the drive when the backup completes.

Backup set will expire: specifies when the backup expires and is no longer restorable.

Initialize and label media: erases and labels media sets. Although this option is available for all tape backups, it is most useful when there are multiple tapes forming a media set.

10. Once all the necessary options are selected, either click the **OK** button to start performing the backup, or check the **Schedule** check box to schedule this operation for periodic execution.

If the backup is performed immediately, the **Backup Progress** dialog box (Figure 10.5) is displayed while the backup is being performed.

Notes

Figure 10.5: Backup Progress Dialog Box

Upon successful completion of the backup, the informational dialog box shown in Figure 10.6 is displayed.

Figure 10.6: Backup Complete Confirmation

The above functionality can be accomplished through Transact-SQL commands executed from SQL Query Analyzer. An example of such a command is illustrated below.

BACKUP DATABASE northwind

TO DISK = 'd:\backups\northwind\nwind.bak'

10.3 Restoring Database

SQL Server supports restoring data on the following levels:

- The database (a complete database restore): The whole database is restored and recovered, and the database is offline for the duration of the restore and recovery operations.
- The data file (a file restore): A data file or a set of files is restored and recovered. During a file restore, the filegroups that contain the files are automatically offline for the duration of the restore. Any attempt to access an offline filegroup causes an error.
- The data page (a page restore): Under the full recovery model or bulk-logged recovery model, you can restore individual databases. Page restores can be performed on any database, regardless of the number of filegroups.

A complete database backup may be restored to the same or a new or different database on the same (or a different) server. The restore operation may be initiated either through either SQL Server Enterprise Manager or the Transact-SQL command window.

It is highly advisable to restore complete database backups at regular intervals, as this is the only means that is currently available to verify the "restorability" of an SQL Server backup.

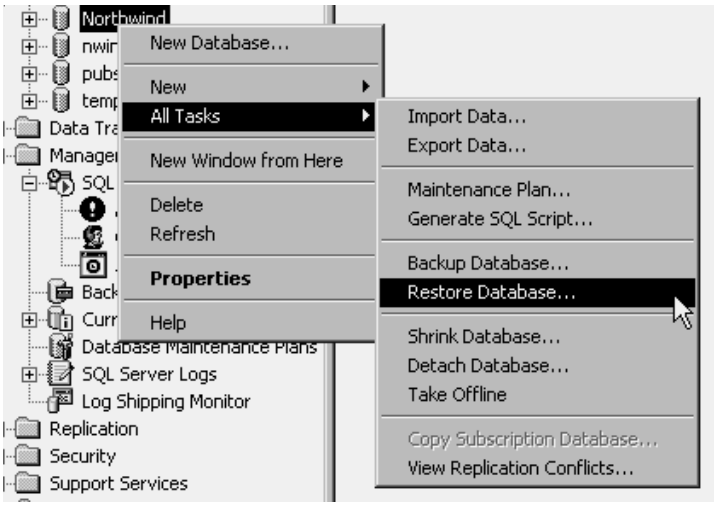
Notes

10.3.1 Restoring a Complete Backup to the Same Database

To restore a complete database backup to the same database follow the given below steps.

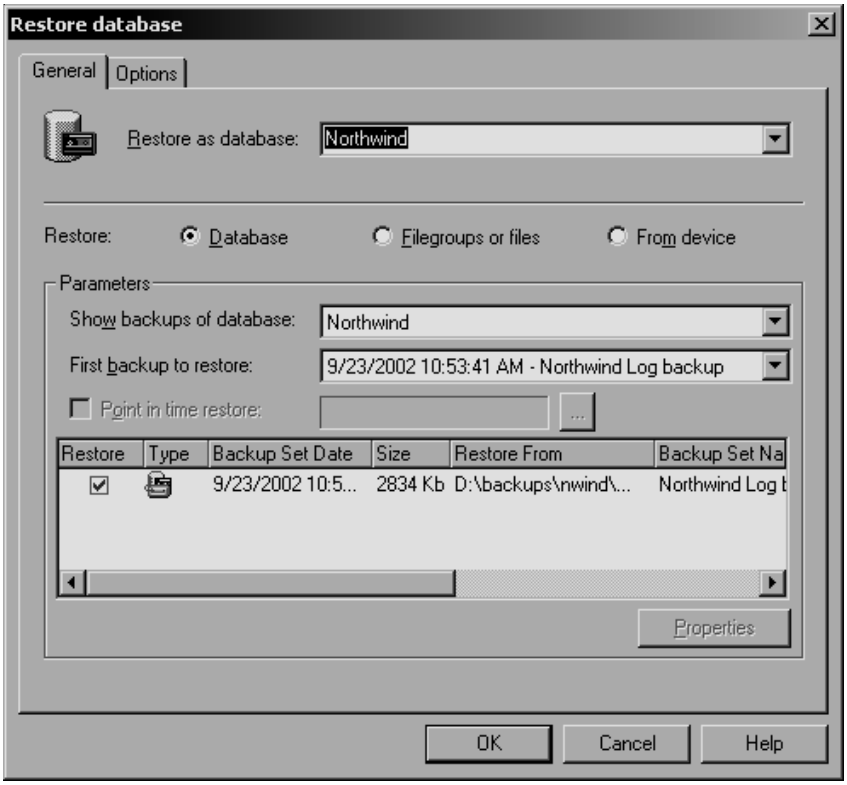
1. Open SQL Server Enterprise Manager and connect to the server where the backup is to be restored.

Figure 10.7: Restore Database Through Enterprise Manager



2. Right-click the database and select **All Tasks**, then select **Restore database** (as shown in Figure 10.8). The **Restore Database** dialog box (Figure 10.8) is displayed.

Figure 10.8: Restore Database Dialog Box



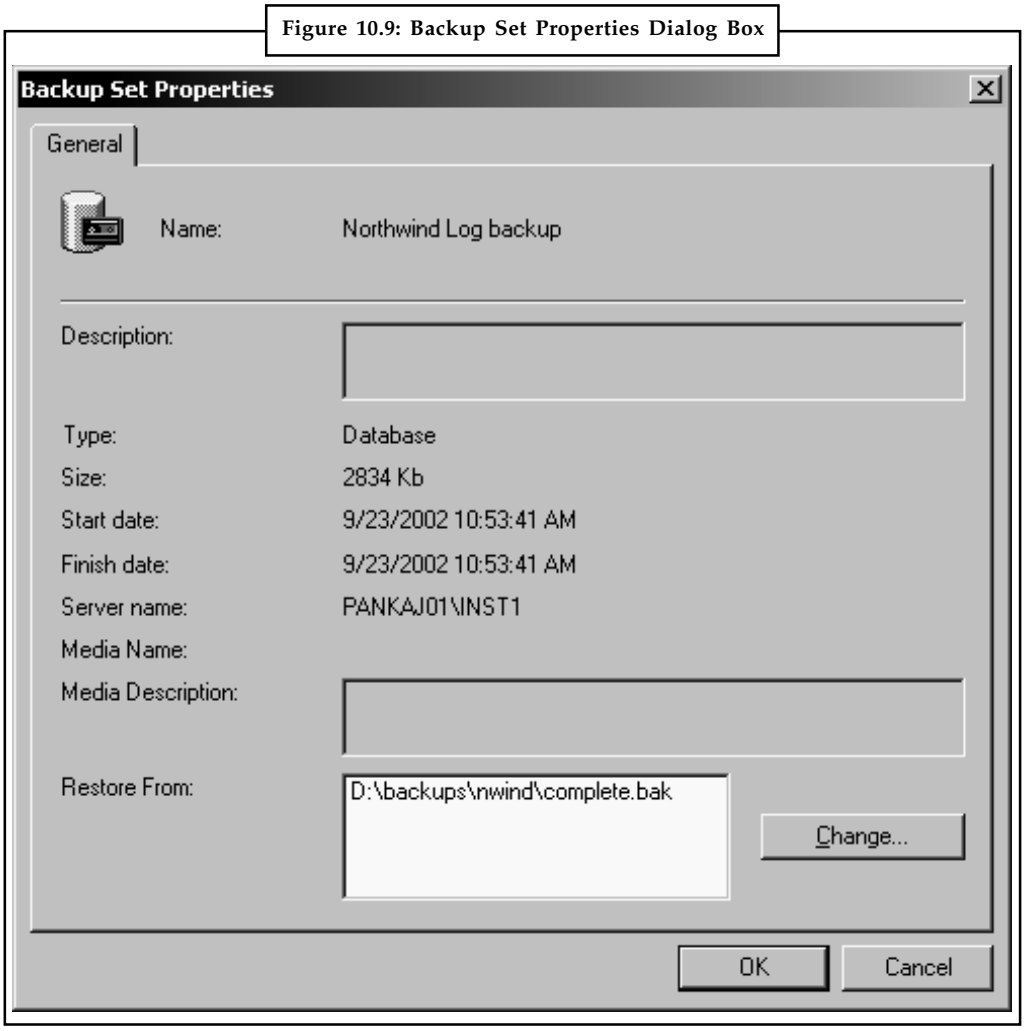
Notes

3. From the list of databases, select the database for which the backup has to be restored.

   A list of all backups performed for the selected database is displayed in the **Parameters** section of the **Restore Database** dialog box. This information is collected from the history tables in the **MSDB** database.

4. From the list of backups, select the backup to restore, then click the **Properties** button to display the **Backup Set Properties** dialog box shown in Figure 10.9.

Figure 10.9: Backup Set Properties Dialog Box



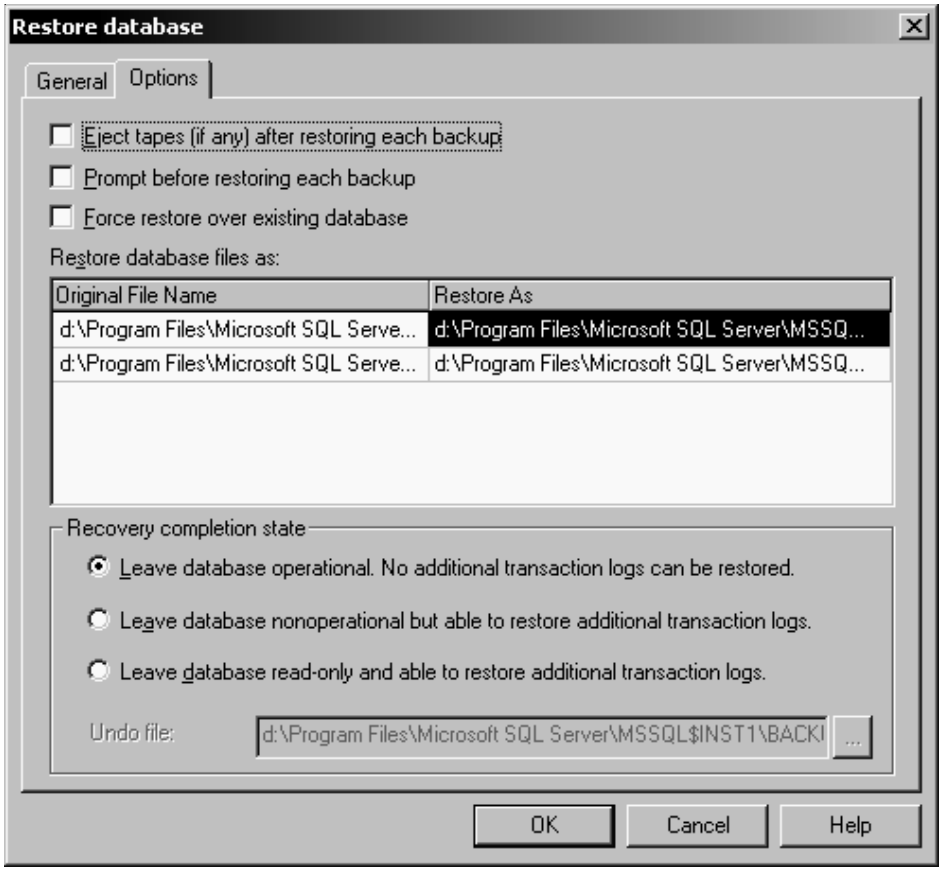
   The dialog box displays backup properties including backup type, size (in KB), start and finish dates, server name, and media description.

5. In the **Backup Set Properties** dialog box, click **OK**.

6. In the **Restore Database** dialog box (Figure 10.10), click the **Options** tab.

Notes

Figure 10.10: Restore Database Options



The **Options** tab in the **Restore Database** dialog box provides options to select the final restore state, change the file name, and set tape options for the restore operation.

7. Select the appropriate settings in this dialog box.

Description of fields

i. ***Eject tapes after restoring each backup:*** ejects the tape when the restore operation completes if a tape restore is being performed.

ii. ***Prompt before restoring each backup:*** displays a dialog box after each backup is restored successfully. This option may be used when restoring multiple backups.

iii. ***Force restore over existing database:*** – forces the files for the existing database to be initialized. This option should be used with caution as it erases the data that exists in the selected database before starting the restore operation.

iv. ***Restore As:*** shows the original and target physical file names for the database that is being restored.

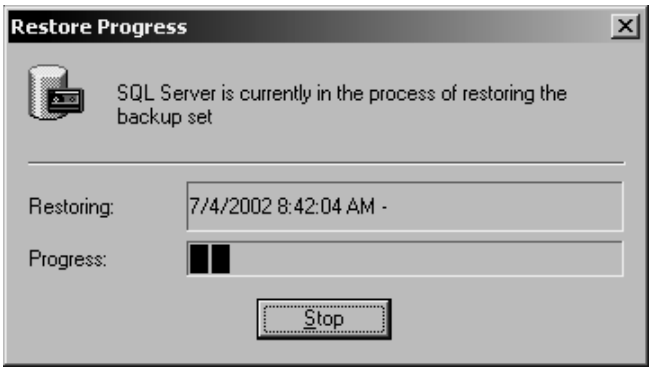
v. ***Recovery completion state:*** determines the final state of the restored database.

vi. ***Undo file:*** a file required by SQL Server to track incomplete transactions if the backup is restored in standby state.

8. To start the restore operation, click **OK**. The **Restore Progress** dialog box (Figure 10.11) is displayed while the restore operation executes.

Notes

Figure 10.11: Restore Progress Dialog Box



9. When the restore operation completes, the dialog box shown in Figure 10.12 is displayed. To close it, click **OK**.

Figure 10.12: Restore Database Complete Confirmation

10.4 Database Snapshots

10.4.1 Meaning of Database Snapshot

A database snapshot is a read-only, static view of a database (the source database). Multiple snapshots can exist on a source database and always reside on the same server instance as the database. Each database snapshot is transactionally consistent with the source database as of the moment of the snapshot's creation. A snapshot persists until it is explicitly dropped by the database owner.

Unlike default behavior for user databases, a database snapshot is created with the ALLOW_SNAPSHOT_ISOLATION database option set ON regardless of the setting of this option on the primary database or the model system database.

Snapshots can be used for reporting purposes. Also, in the event of a user error on a source database, you can revert the source database to the state it was in when the snapshot was created. Data loss is confined to updates to the database since the snapshot's creation.

Caution Reverting does not work in an offline or corrupted database. Therefore, taking regular backups and testing your restore plan are necessary to protect a database.

Notes Database snapshots are unrelated to snapshot backups, snapshot isolation of transactions, or snapshot replication.

Notes

10.4.2 How Database Snapshots Work

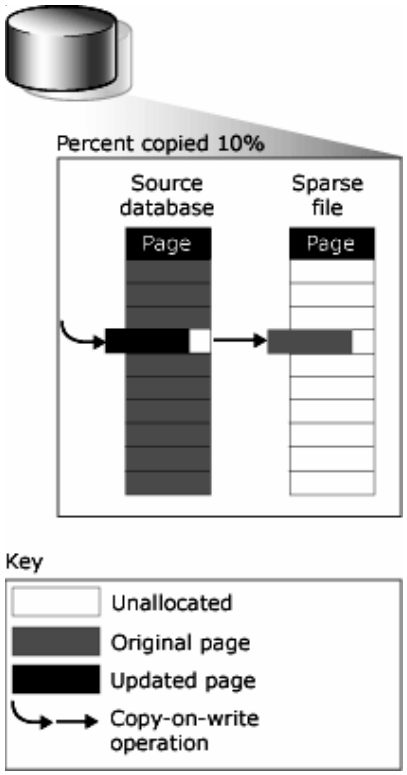
A database snapshot provides a read-only, static view of a source database as it existed at snapshot creation, minus any uncommitted transactions. Uncommitted transactions are rolled back in a newly created database snapshot because the Database Engine runs recovery after the snapshot has been created (transactions in the database are not affected).

Database snapshots are dependent on the source database. The snapshots of a database must be on the same server instance as the database. Furthermore, if that database becomes unavailable for any reason, all of its database snapshots also become unavailable.

Snapshots can be used for reporting purposes. Also, in the event of a user error on a source database, you can revert the source database to the state it was in when the snapshot was created. Data loss is confined to updates to the database since the snapshot's creation. Also, creating a database snapshot can be useful immediately before making a major change to a database, such as changing the schema or the structure of a table.

Understanding how snapshots work is helpful though not essential to using them. Database snapshots operate at the data-page level. Before a page of the source database is modified for the first time, the original page is copied from the source database to the snapshot. This process is called a copy-on-write operation. The snapshot stores the original page, preserving the data records as they existed when the snapshot was created. Subsequent updates to records in a modified page do not affect the contents of the snapshot. The same process is repeated for every page that is being modified for the first time. In this way, the snapshot preserves the original pages for all data records that have ever been modified since the snapshot was taken.

Figure 10.13: Copy-on Write Operation



Notes

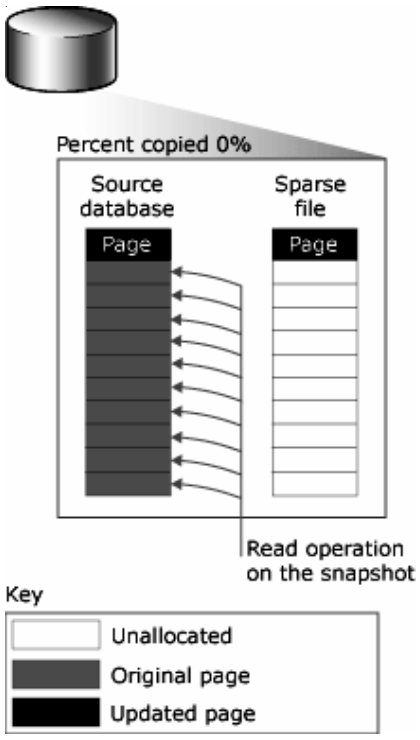
To store the copied original pages, the snapshot uses one or more sparse files. Initially, a sparse file is an essentially empty file that contains no user data and has not yet been allocated disk space for user data. As more and more pages are updated in the source database, the size of the file grows. When a snapshot is taken, the sparse file takes up little disk space. As the database is updated over time, however, a sparse file can grow into a very large file.

The following figure illustrates a copy-on-write operation. The light gray rectangles in the snapshot diagram represent potential space in a sparse file that is as-yet unallocated. On receiving the first update to a page in the source database, the Database Engine writes to the file and the operating system allocates space in the snapshot's sparse files and copies the original page there. The Database Engine then updates the page in the source database. The following figure 10.13 illustrates such a copy-on-write operation.

Did u know? Because database snapshots are not redundant storage, they do not protect against disk errors or other types of corruption. Taking regular backups and testing your restore plan are essential to protect a database. If you must restore the source database to the point in time at which you created a database snapshot, implement a backup policy that enables you to do that.

To the user, a database snapshot appears never to change, because read operations on a database snapshot always access the original data pages, regardless of where they reside. If the page has not yet been updated on the source database, a read operation on the snapshot reads the original page from the source database. The following figure shows a read operation on a newly created snapshot, whose sparse file accordingly contains no pages. This read operation reads only from the source database.

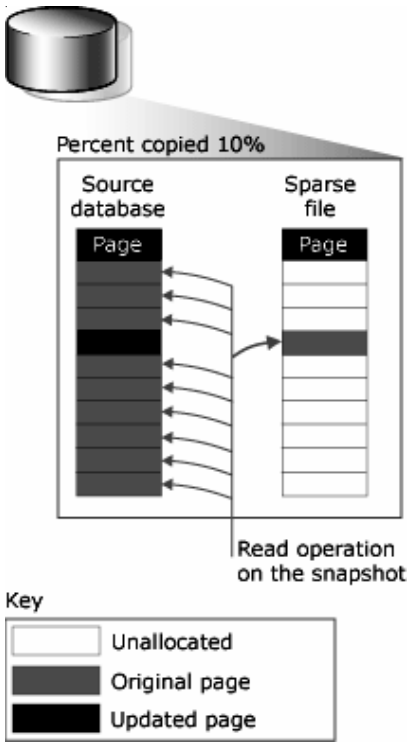
Figure 10.14: Read Operation



Notes

After a page has been updated, a read operation on the snapshot still accesses the original page, which is now stored in a sparse file. The following figure illustrates a read operation on the snapshot that accesses a page after it has been updated in the source database. The read operation reads the original page from the sparse file of the snapshot.

Figure 10.15: Read Operation



Self Assessment

State true or false:

6. Database snapshots are independent on the source database
7. Database snapshots operate at the data-page level.
8. Taking regular backups and testing your restore plan are essential to protect a database.
9. The snapshots of a database must be on different server instance as the database.
10. To the user, a database snapshot appears never to change, because read operations on a database snapshot always access the original data pages, regardless of where they reside.

10.4.3 Effect of the Update Pattern on Database Snapshot Growth

If your source database is fairly large and you are concerned about disk space usage, at some point you should replace an old snapshot with a new snapshot. The ideal lifespan of a snapshot depends on its growth rate and the disk space that is available to its sparse files. The disk space required by a snapshot depends on how many different pages in the source database are updated during the life of the snapshot. Therefore, if updates are mostly to a small subset of pages that are updated repeatedly, the growth rate will slow over time and the snapshot space requirements will remain relatively small. In contrast, when all of the original pages are eventually updated

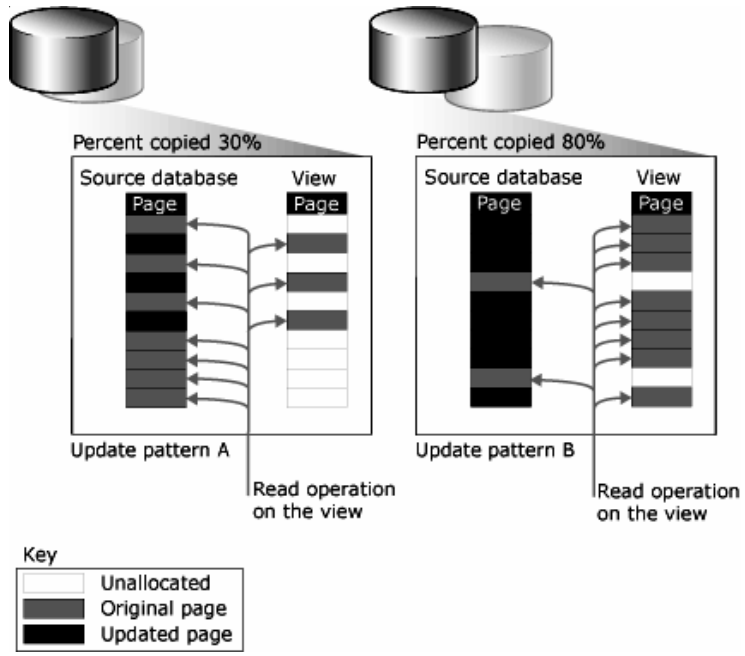
Notes

at least once, the snapshot will grow to the size of the source database. If the disk begins to fill up, the snapshots compete with each other for disk space. If the disk drive fills up, write operations to all the snapshots will fail.

Therefore, it is useful to know the typical update patterns for a database when planning how much space is required during the planned lifespan of a snapshot. For some databases, the rate of updates may be fairly constant; for example, an inventory database might have many of its pages updated daily, making it useful to replace old snapshots daily or weekly. For other databases, the proportion of updated pages may vary during the business cycle; for example, a catalog database might be updated primarily quarterly, with only occasional updates at other times; creating snapshots just before and after each quarterly update would be a logical strategy. The pre-update snapshot would permit reverting if a significant update error occurs, and the post-update snapshot could be used for report writing during the next quarter.

The following figure illustrates the effects of two contrasting update patterns on the size of a snapshot. Update pattern A reflects an environment in which only 30 percent of the original pages are updated during the life of the snapshot. Update pattern B reflects an environment in which 80 percent of the original pages are updated during the life of the snapshot.

Figure 10.16: Effects of Two Contrasting update Patterns on the Size of a Snapshot



10.4.4 Metadata about Database Snapshots

For database snapshots, database metadata includes the **source_database_id** property, which is stored in a column of the **sys.databases** catalog view. Generally, a database snapshot does not expose metadata of its own, but it does expose metadata from its source database. This metadata includes, for example, the data returned by the following statement:

USE <database_snapshot> SELECT * FROM sys.database_files

where <database_snapshot>is the name of a database snapshot.

The only exceptions are when the source database uses full-text search or database mirroring, which disable themselves on a snapshot by altering some values in the snapshot's metadata.

Notes

10.4.5 Typical Uses of Database Snapshots

A database snapshot is a read-only, static view of a database (called the source database). Each database snapshot is transactionally consistent with the source database at the moment of the snapshot's creation. When you create a database snapshot, the source database will typically have open transactions. Before the snapshot becomes available, the open transactions are rolled back to make the database snapshot transactionally consistent.

Clients can query a database snapshot, which makes it useful for writing reports based on the data at the time of snapshot creation. Also, if the source database later becomes damaged, you can revert the source database to the state it was in when the snapshot was created.

Notes Database snapshots are available only in SQL Server Enterprise.

10.4.6 Reasons to take Database Snapshots

It include:

- Maintaining historical data for report generation.

  Because a database snapshot provides a static view of a database, a snapshot can extend access to data from a particular point in time. For example, you can create a database snapshot at the end of a given time period (such as a financial quarter) for later reporting. You can then run end-of-period reports on the snapshot. If disk space permits, you can also maintain end-of-period snapshots indefinitely, allowing queries against the results from these periods; for example, to investigate organizational performance.

- Using a mirror database that you are maintaining for availability purposes to off-load reporting.

  Using database snapshots with database mirroring permits you to make the data on the mirror server accessible for reporting. Additionally, running queries on the mirror database can free up resources on the principal. Safeguarding data against administrative error.

- Before doing major updates, such as a bulk update or a schema change, create a database snapshot on the database protects data. If you make a mistake, you can use the snapshot to recover by reverting the database to the snapshot. Reverting is potentially much faster for this purpose than restoring from a backup; however, you cannot roll forward afterward.

Notes Database snapshots are dependent on the source database. Therefore, using database snapshots for reverting a database is not a substitute for your backup and restore strategy. Performing all your scheduled backups remains essential. If you must restore the source database to the point in time at which you created a database snapshot, implement a backup policy that enables you to do that.

- Safeguarding data against user error.

  By creating database snapshots on a regular basis, you can mitigate the impact of a major user error, such as a dropped table. For a high level of protection, you can create a series of database snapshots spanning enough time to recognize and respond to most user errors. For instance, you might maintain 6 to 12 rolling snapshots spanning a 24-hour interval,

depending on your disk resources. Then, each time a new snapshot is created, the earliest snapshot can be deleted.

- To recover from a user error, you can revert the database to the snapshot immediately before the error. Reverting is potentially much faster for this purpose than restoring from a backup; however, you cannot roll forward afterward.
- Alternatively, you may be able to manually reconstruct a dropped table or other lost data from the information in a snapshot. For instance, you could bulk copy the data out of the snapshot into the database and manually merge the data back into the database.

Notes Your reasons for using database snapshots determine how many concurrent snapshots you need on a database, how frequently to create a new snapshot, and how long to keep it.

- Managing a test database

  In a testing environment, it can be useful when repeatedly running a test protocol for the database to contain identical data at the start of each round of testing. Before running the first round, an application developer or tester can create a database snapshot on the test database. After each test run, the database can be quickly returned to its prior state by reverting the database snapshot.

10.4.7 Limitations and Requirements of Database Snapshots

A database snapshot captures the point in time at which snapshot creation began, minus any uncommitted transactions. Before using database snapshots, you should understand the impact of database snapshots on the source database and the system environment, as well as limitations on the snapshots themselves.

Limitations on the Source Database

As long as a database snapshot exists, the following limitations exist on the snapshot's source database:

- The database cannot be dropped, detached, or restored.

Notes Backing up the source database works normally; it is unaffected by database snapshots.

- Performance is reduced, due to increased I/O on the source database resulting from a copy-on-write operation to the snapshot every time a page is updated.
- Files cannot be dropped from the source database or from any snapshots.
- The source database must be online, unless the database is a mirror database within a database mirroring session.

Notes All recovery models support database snapshots.

Notes

- The source database cannot be configured as a scalable shared database.
- To create a database snapshot on a mirror database, the database must be in the synchronized mirroring state.

Limitations on Database Snapshots

The following limitations apply to database snapshots:

- A database snapshot must be created and remain on the same server instance as the source database.
- Database snapshots always work on an entire database.
- Database snapshots are dependent on the source database. Therefore, using database snapshots for reverting a database is not a substitute for your backup and restore strategy. Performing all your scheduled backups remains essential. If you must restore the source database to the point in time at which you created a database snapshot, implement a backup policy that enables you to do that.
- When a page getting updated on the source database is pushed to a snapshot, if the snapshot runs out of disk space or encounters some other error, the snapshot becomes suspect and must be deleted.
- Snapshots are read-only.
- Snapshots of the **model**, **master**, and **tempdb** databases are prohibited.
- You cannot change any of the specifications of the database snapshot files.
- You cannot drop files from a database snapshot.
- You cannot back up or restore database snapshots.
- You cannot attach or detach database snapshots.
- You cannot create database snapshots on FAT32 file system or RAW partitions. The sparse files used by database snapshots are provided by the NTFS file system.
- Full-text indexing is not supported on database snapshots. Full-text catalogs are not propagated from the source database.
- A database snapshot inherits the security constraints of its source database at the time of snapshot creation. Because snapshots are read-only, inherited permissions cannot be changed and permission changes made to the source will not be reflected in existing snapshots.
- A snapshot always reflects the state of filegroups at the time of snapshot creation: online filegroups remain online, and offline filegroups remain offline. For more information, see "Database Snapshots with Offline Filegroups" later in this topic.
- If a source database becomes RECOVERY_PENDING, its database snapshots may become inaccessible. After the issue on the source database is resolved, however, its snapshots should become available again.
- Reverting is unsupported for read-only filegroups and for compressed filegroups. Attempts to revert a database containing either of these types of filegroups fail.
- In a log shipping configuration, database snapshots can be created only on the primary database, not on a secondary database. If you switch roles between the primary server instance and a secondary server instance, you must drop all the database snapshots before you can set the primary database up as a secondary database.

Notes

- A database snapshot cannot be configured as a scalable shared database.
- FILESTREAM filegroups are not supported by database snapshots. If FILESTREAM filegroups exist in a source database, they are marked as offline in its database snapshots, and the database snapshots cannot be used for reverting the database.

Caution A SELECT statement that is executed on a database snapshot must not specify a FILESTREAM column; otherwise, the following error message will be returned: Could not continue scan with NOLOCK due to data movement.

Disk Space Requirements

Database snapshots consume disk space. If a database snapshot runs out of disk space, it is marked as suspect and must be dropped. (The source database, however, is not affected; actions on it continue normally.) Compared to a full copy of a database, however, snapshots are highly space efficient. A snapshot requires only enough storage for the pages that change during its lifetime. Generally, snapshots are kept for a limited time, so their size is not a major concern.

The longer you keep a snapshot, however, the more likely it is to use up available space. The maximum size to which a sparse file can grow is the size of the corresponding source database file at the time of the snapshot creation. If a database snapshot runs out of disk space, it must be deleted (dropped).

Notes Except for file space, a database snapshot consumes roughly as many resources as a database.

10.4.8 Creating a Database Snapshot

This topic describes some best practices for creating database snapshots. Following are some best practices for naming database snapshots, timing when you create them, limiting their number, and redirecting client connections to a snapshot.

Naming Database Snapshots

Before creating snapshots, it is important to consider how to name them. Each database snapshot requires a unique database name. For administrative ease, the name of a snapshot can incorporate information that identifies the database, such as:

- The name of the source database.
- An indication that the new name is for a snapshot.
- The creation date and time of the snapshot, a sequence number, or some other information, such as time of day, to distinguish sequential snapshots on a given database.

For example, consider a series of snapshots for the AdventureWorks2008R2 database. Three daily snapshots are created at 6-hour intervals between 6 A.M. and 6 P.M., based on a 24-hour clock. Each daily snapshot is kept for 24 hours before being dropped and replaced by a new snapshot of the same name. Note that each snapshot name indicates the hour, but not the day:

Notes

AdventureWorks2008R2_snapshot_0600

AdventureWorks2008R2_snapshot_1200

AdventureWorks2008R2_snapshot_1800

Alternatively, if the creation time of these daily snapshots varies from day to day, a less precise naming convention might be preferable, for example:

AdventureWorks2008R2_snapshot_morning

AdventureWorks2008R2_snapshot_noon

AdventureWorks2008R2_snapshot_evening

Limiting the Number of Database Snapshots

Creating a series of snapshots over time captures sequential snapshots of the source database. Each snapshot persists until it is explicitly dropped. Because each snapshot will continue to grow as original pages are updated, you may want to conserve disk space by deleting an older snapshot after creating a new snapshot.

Notes If you want to revert to a database snapshot, you need to delete any other snapshots from that database.

Client Connections to a Database Snapshot

To use a database snapshot, clients need to know where to find it. Users can read from one database snapshot while another is being created or deleted. However, when you substitute a new snapshot for an existing one, you need to redirect clients to the new snapshot. Users can manually connect to a database snapshot by means of SQL Server Management Studio. However, to support a production environment, you should create a programmatic solution that transparently directs report-writing clients to the latest database snapshot of the database.

Did u know? SQL Server Management Studio does not support the creation of database snapshots.

10.5 Summary

- A copy of data that can be used to restore and recover the data is called a backup. Backups let you restore data after a failure. With good backups, you can recover from many failures.
- The scope of a backup of data (a data backup) can be a whole database, a partial database, or a set of files or filegroups. For each of these, SQL Server supports full and differential backups
- Database backups are easy to use and are recommended whenever database size allows. SQL Server supports the following types of database backups such as full backups, partial backup, file backup etc.
- SQL Server supports restoring data on the following levels:
- The database (a complete database restore): The whole database is restored and recovered, and the database is offline for the duration of the restore and recovery operations.

Notes

- The data file (a file restore): A data file or a set of files is restored and recovered. During a file restore, the filegroups that contain the files are automatically offline for the duration of the restore. Any attempt to access an offline filegroup causes an error.
- A database snapshot is a read-only, static view of a database (the source database). Database snapshots are dependent on the source database. The snapshots of a database must be on the same server instance as the database. Database snapshots operate at the data-page level. To the user, a database snapshot appears never to change, because read operations on a database snapshot always access the original data pages, regardless of where they reside.

10.6 Keywords

Database Snapshot: A database snapshot is a read-only, static view of a database (the source database).

Differential Backup: A differential backup is based on the latest full backup of the data. This is known as the base of the differential, or the differential base. A differential base is a full backup of read/write data.

Differential File Backup: A backup of one or more files that contain data extents that were changed since the most recent full backup of each file.

Differential Partial Backup: A backup that contains only the data extents that were modified since the most recent partial backup of the same set of filegroups.

File Backup: A full backup of all the data in one or more files, or filegroups.

Full Backup: A full backup contains all the data in a specific database or set of filegroups or files, and also enough log to allow for recovering that data.

Partial Backup: A backup of all the full data in the primary filegroup, every read/write filegroup, and any optionally specified read-only files or filegroups. A partial backup of a read-only database contains only the primary filegroup.

10.7 Review Questions

1. Briefly describe various types of backup strategies
2. Discuss the back media available in SQL server.
3. Explain the concept of database snapshot.
4. Describe the advantages and disadvantages of database snapshot.
5. Explain the basic uses of database snapshot.
6. Comment on the following statement: Effect of the Update Pattern on Database Snapshot Growth.
7. Describe the process of creating database snapshots.
8. Explain in detail the procedure to perform a complete database backup through SQL Server Enterprise Manager.
9. Discuss how to restore a complete database backup to same database.

Answers: Self Assessment

1. Full backup
2. Differential backup
3. Partial Backup
4. Differential Partial Backup

Notes

5.	File Backup	6.	False
7.	True	8.	True
9.	False	10.	True

10.8 Further Readings

Books

C.J. Date, *Introduction to Database Systems*, Pearson Education.

Elmasri Navrate, *Fundamentals of Database Systems*, Pearson Education.

Peter Rob & Carlos Coronel, *Database Systems Design, Implementation and Management*, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, *Database Management Systems*, Tata McGraw Hill, 3rd Edition.

Silberschatz, Korth, *Database System Concepts*, McGraw Hill, 5th Edition.

Online links

www.sql-tutorial.net/

www.w3schools.com/sql/

www.sqlservertutorials.com/

Unit 11: Monitoring SQL Server for Performance

Notes

CONTENTS

Objectives

Introduction

11.1 Need of Monitoring

11.2 Monitoring Process

11.3 Evaluating Performance

11.4 Factors Affecting Performance

11.5 Tools and Techniques for Monitoring

11.6 Choosing a Monitoring Tool

11.7 Summary

11.8 Keywords

11.9 Review Questions

11.10 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand the need of monitoring in a SQL server
- Conceptualize the monitoring process
- Learn about different criteria for evaluating performance
- Know how to choosing a monitoring tool in SQL server

Introduction

Microsoft® SQL Server™ 2000 provides a variety of tools that can be used to monitor the performance of an instance of SQL Server and the user activity that occurs in databases. Monitoring allows you to determine whether your database application is working efficiently and as expected, even as your application, database, and environment change. For example, as more concurrent users use a database application, the load on SQL Server can increase. By monitoring, you can determine whether the current instance of SQL Server or system configuration must be changed to handle the increased workload, or whether the increased load is having no significant effect on performance.

11.1 Need of Monitoring

Monitoring SQL Server allows you to:

- ***Determine whether it is possible to improve performance.*** For example, by monitoring the response times for frequently used queries, you can determine whether changes to the query or indexes on the tables are necessary.

Notes

- ***Evaluate user activity.*** For example, by monitoring users attempting to connect to an instance of SQL Server, you can determine whether security is set up adequately and test applications and development systems. For example, by monitoring SQL queries as they are executed, you can determine whether they are written correctly and producing the expected results.
- Troubleshoot any problems or debug application components, such as stored procedures.

11.2 Monitoring Process

Monitoring process consists of a set of procedures to monitor an application, an instance of SQL Server, or the operating system environment (hardware and software). These Steps have been briefly discussed below:

1. Determine your monitoring goals.
2. Choose the appropriate tool for the type of monitoring you will perform.
3. Use the tool to monitor SQL Server or the system environment and analyze the captured data.
4. ***Identify the events to monitor:*** The events determine which activities are monitored and captured. Your selection of events to monitor will depend on what is being monitored and why. For example, when monitoring disk activity, it is not necessary to monitor SQL Server locks.
5. ***Determine the event data to capture:*** The event data describes each instance of an event as it occurs. For example, when monitoring lock events, you can capture data describing the tables, users, and connections affected by the lock event. The following explains the process involved in capturing event data and putting it to use.

   i. *Apply filters to limit the event data collected.* Limiting the event data allows the system to focus on the events pertinent to the monitoring scenario. For example, if you want to monitor slow queries, you can use a filter to monitor only those queries issued by the application that take more than 30 seconds to execute against a particular database.

   ii. *Monitor (capture) events.* Once enabled, active monitoring captures data from the specified application, instance of SQL Server, or operating system. For example, when disk activity is monitored using System Monitor (Performance Monitor in Microsoft Windows NT® 4.0), monitoring captures event data such as disk reads and writes and displays it to the screen.

   iii. *Save captured event data.* Saving captured data allows you to analyze it at a later time or even replay it using SQL Profiler. Captured event data is saved to a file that can be loaded back into the tool that originally created the file for analysis. SQL Profiler allows event data to be saved to a SQL Server table. Saving captured event data is vital when creating a performance baseline. The performance baseline data is saved and used when comparing recently captured event data to determine whether performance is optimal.

   iv. *Create definition files containing the settings specified to capture the events.* Definition files include specifications about the events themselves, event data, and filters that are used to capture data. These files can be used to monitor a specific set of events at a later time without redefining the events, event data, and filters. For example, if you want to monitor frequently the number of deadlocks and the users involved in

Notes

those deadlocks, you can create a file defining those events, event data, and event filters; save the definition; and reapply the filter the next time you want to monitor deadlocks. SQL Profiler uses trace definition files for this purpose.

v. *Analyze captured event data.* In order to be analyzed, the captured, saved event data is loaded into the application that captured the data. For example, a captured trace from SQL Profiler can be reloaded into SQL Profiler for viewing and analysis. Analyzing event data involves determining what is happening and why. This information allows you to make changes that can improve performance, such as adding more memory, changing indexes, correcting coding problems with Transact-SQL statements or stored procedures, and so on, depending on the type of analysis performed. For example, you can use the Index Tuning Wizard to analyze a captured trace from SQL Profiler automatically and make index recommendations based on the results.

vi. *Replay captured event data.* Only available in SQL Profiler, event replay allows you to establish a test copy of the database environment from which the data was captured and repeat the captured events as they occurred originally on the real system. You can replay them at the same speed as they originally occurred, as fast as possible (to stress the system), or more likely, one step at a time (to analyze the system after each event has occurred). By analyzing the exact events in a test environment, you can prevent detrimental effects on the production system.

Self Assessment

State true or false:

1. SQL Profiler allows event data to be saved to a SQL event.
2. Event replay option is only available in SQL Profiler only.
3. Limiting the event data allows the system to focus on the events pertinent to the monitoring scenario.
4. Saving captured event data is vital when creating a performance baseline.
5. The events determine which activities are monitored and captured.

11.3 Evaluating Performance

Optimal performance comes from minimal response times and maximum throughput as a result of efficient network traffic, disk I/O, and CPU time. This goal is achieved by analyzing thoroughly the application requirements, understanding the logical and physical structure of the data, and assessing and negotiating tradeoffs between conflicting uses of the database, such as online transaction processing (OLTP) versus decision support.

Response Time vs. Throughput

Response time is measured as the length of time required for the first row of the result set to be returned to the user in the form of visual confirmation that a query is being processed.

Throughput is a measure of the total number of queries handled by the server during a given time.

As the number of users increases, so does the competition for a server's resources, which in turn causes response time to increase and overall throughput to decrease.

Notes

11.4 Factors Affecting Performance

The following areas affect the performance of SQL Server:

- System resources (hardware)
- The Microsoft Windows NT® 4.0 and Windows® 2000 operating systems
- Database applications
- Client applications
- Network

Before these areas can be monitored, you must know what level of performance is reasonable given normal working conditions. To do this, establish a server performance baseline by monitoring Microsoft® SQL Server™ performance at regular intervals, even when no problems occur.

Troubleshooting Problems

You can monitor the following areas to troubleshoot problems:

- SQL Server stored procedures or batches of SQL statements submitted by user applications.
- User activity, such as blocking locks or deadlocks.
- Hardware activity, such as disk usage

11.5 Tools and Techniques for Monitoring

There are some Microsoft tools that can help monitor your database performance:

1. ***SQL Profiler:*** SQL Profiler enables you to monitor server and database activity (for example, number of deadlocks, fatal errors, tracing stored procedures and Transact-SQL statements, or login activity). You can capture SQL Profiler data to a SQL Server table or a file for later analysis, and also replay the events captured on SQL Server, step by step, to see exactly what happened. SQL Profiler tracks engine process events, such as the start of a batch or a transaction.

2. ***System Monitor:*** System Monitor enables you to monitor server performance and activity using predefined objects and counters or user-defined counters to monitor events. System Monitor (Performance Monitor in Microsoft Windows NT® 4.0) collects counts rather than data about the events (for example, memory usage, number of active transactions, number of blocked locks, or CPU activity). You can set thresholds on specific counters to generate alerts that notify operators. System Monitor primarily tracks resource usage, such as the number of buffer manager page requests in use.

   System Monitor works only on Microsoft Windows® 2000 and can monitor (remotely or locally) an instance of SQL Server on Windows NT 4.0 or Windows 2000 only.

3. ***Current activity window (SQL Server Enterprise Manager):*** Graphically displays information about processes running currently on an instance of SQL Server, blocked processes, locks, and user activity. This is useful for ad hoc views of current activity.

4. ***Error Logs:*** Contain additional information about events in SQL Server than is available elsewhere. You can use the information in the error log to troubleshoot SQL Server-related problems. The Windows application event log provides an overall picture of

Notes

events occurring on the Windows NT 4.0 and Windows 2000 system as a whole, as well as events in SQL Server, SQL Server Agent, and full-text search.

5. ***sp_who:*** Reports snapshot information about current SQL Server users and processes, including the currently executing statement and whether the statement is blocked. This is a Transact-SQL alternative to viewing user activity in the current activity window in SQL Server Enterprise Manager.

6. ***sp_lock:*** Reports snapshot information about locks, including the object ID, index ID, type of lock, and type or resource to which the lock applies. This is a Transact-SQL alternative to viewing lock activity in the current activity window in SQL Server Enterprise Manager.

7. ***sp_spaceused:*** Displays an estimate of the current amount of disk space used by a table (or a whole database). This is a Transact-SQL alternative to viewing database usage in SQL Server Enterprise Manager.

8. ***sp_monitor:*** Displays statistics, including CPU usage, I/O usage, and the amount of time idle since **sp_monitor** was last executed.

9. ***DBCC statements:*** Enables you to check performance statistics and the logical and physical consistency of a database.

10. ***Built-in functions:*** Display snapshot statistics about SQL Server activity since the server was started; these statistics are stored in predefined SQL Server counters. For example, @@CPU_BUSY contains the amount of time the CPU has been executing SQL Server code; @@CONNECTIONS contains the number of SQL Server connections or attempted connections; and @@PACKET_ERRORS contains the number of network packets occurring on SQL Server connections.

11. ***SQL Profiler stored procedures and functions:*** Use Transact-SQL stored procedures to gather SQL Profiler statistics.

12. ***Trace flags:*** Display information about a specific activity within the server and are used to diagnose problems or performance issues (for example, deadlock chains).

13. ***Simple Network Management Protocol (SNMP):*** Simple Network Management Protocol (SNMP) is an application protocol that offers network management services. Using SNMP, you can monitor an instance of SQL Server across different platforms (for example, Windows NT 4.0, Windows 98, and UNIX). With SQL Server and the Microsoft SQL Server Management Information Base (MSSQL-MIB), you can use SNMP applications to monitor the status of SQL Server installations. You can monitor performance information, access databases, and view server and database configuration parameters.

We will study about all these tools for monitoring performance in the detail in the next unit.

Task Name any five monitoring tools and list its main functions

Using SQL Profiler System Stored Procedures

The key difference between the two main monitoring tools, SQL Profiler and System Monitor, is that SQL Profiler monitors engine events while System Monitor monitors resource usage associated with server processes. For example, SQL Profiler can be used to monitor deadlocks events, including the users and objects involved in the deadlock. System Monitor can be used to monitor the total number of deadlocks occurring in a database or on a specific object.

Notes

Windows NT 4.0 and Windows 2000 also provides these monitoring tools:

- ***Task Manager:*** Shows a synopsis of the processes and applications running on the system.
- ***Network Monitor Agent:*** Assists in monitoring network traffic.
- There are also some **third-party tools** such as BMC DBXray, Embarcadero Performance Center, HybridX SQL Spy, Intrinsic Design Coefficient, Mercury Interactive SiteScope, NetIQ DiagnosticManager, Zero Impact SQL Monitor, NetIQ AppManager, SQL Server Monitor, etc.

Caution Keep in mind that you probably don't want these products running against your SQL Server machine 24x7, since they may take resources away and actually contribute to the problem

11.6 Choosing a Monitoring Tool

Microsoft® SQL Server™ provides a comprehensive set of tools for monitoring events in SQL Server. Your choice of tool will depend on the type of monitoring and the events to be monitored. For example, ad hoc monitoring to determine the number of users currently connected to an instance of SQL Server can be accomplished by using the **sp_who** system stored procedure, rather than creating a trace and using SQL Profiler.

The choice of a monitoring tool depends on the type of events and activity to be monitored.

Event or activity	SQL Profiler	System Monitor	Current Activity Window	Transact-SQL	Error logs
Trend analysis	Yes	Yes			
Replaying captured events	Yes				
Ad hoc monitoring	Yes		Yes	Yes	Yes
Generating alerts		Yes			
Graphical interface	Yes	Yes	Yes		Yes
Using within custom application	Yes 1			Yes	

Self Assessment

Give one word for the following:

6. Time required for the first row of the result set to be returned to the user in the form of visual confirmation that a query is being processed.
7. A time measure of the total number of queries handled by the server during a given time.
8. A monitoring tool that enables you to monitor server and database activity
9. A monitoring tool that enables you to monitor server performance and activity using predefined objects and counters or user-defined counters.
10. A windows monitoring tool that shows a synopsis of the processes and applications running on the system.
11. A tool which enables you to check performance statistics and the logical and physical consistency of a database.

Notes

11.7 Summary

- Monitoring process consists of a set of procedures to monitor an application, an instance of SQL Server, or the operating system environment (hardware and software).
- Optimal performance comes from minimal response times and maximum throughput as a result of efficient network traffic, disk I/O, and CPU time.
- The following areas affect the performance of SQL Server:
  - System resources (hardware)
  - The Microsoft Windows NT® 4.0 and Windows® 2000 operating systems
  - Database applications
  - Client applications
  - Network
- There are some Microsoft tools that can help monitor your database performance such as SQL Profiler, system monitor, error log, Current activity window, sp_who, sp_lock, sp_spaceused, sp_monitor, DBCC statements, etc.
- Microsoft® SQL Server provides a comprehensive set of tools for monitoring events in SQL Server. Your choice of tool will depend on the type of monitoring and the events to be monitored.

11.8 Keywords

Monitoring Process: Monitoring process consists of a set of procedures to monitor an application, an instance of SQL Server, or the operating system environment (hardware and software).

Response Time: Response time is measured as the length of time required for the first row of the result set to be returned to the user in the form of visual confirmation that a query is being processed.

SQL Profiler: SQL Profiler enables you to monitor server and database activity

System Monitor: System Monitor enables you to monitor server performance and activity using predefined objects and counters or user-defined counters to monitor events.

Throughput: Throughput is a measure of the total number of queries handled by the server during a given time.

11.9 Review Questions

1. Why do you need to monitor performance of sql server?
2. What are the various factors which affect the performance of SQL server?
3. Explain in detail the monitoring process of sql server.
4. What are the different criteria for evaluating performance?
5. Explain how would you a choose a monitoring tool in SQL server?
6. Write short notes on following:
   (a) Simple Network Management Protocol (SNMP)
   (b) Built-in functions

Notes

(c) Error Logs

(d) Current activity window (SQL Server Enterprise Manager)

(e) System Monitor

(f) SQL Profiler

Answers: Self Assessment

1.	False	2.	False
3.	True	4.	True
5.	True	6.	Response time
7.	Throughput	8.	SQL Profiler
9.	System Monitor	10.	Task Manager
11.	DBCC Statements		

11.10 Further Readings

Books

C.J. Date, *Introduction to Database Systems*, Pearson Education.

Elmasri Navrate, *Fundamentals of Database Systems*, Pearson Education.

Peter Rob & Carlos Coronel, *Database Systems Design, Implementation and Management*, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, *Database Management Systems*, Tata McGraw Hill, 3rd Edition.

Silberschatz, Korth, *Database System Concepts*, McGraw Hill, 5th Edition.

Online links

www.sql-tutorial.net/

www.w3schools.com/sql/

www.sqlservertutorials.com/

Notes

Unit 12: Tools and Techniques for Monitoring Performance

CONTENTS

Objectives
Introduction
12.1 Benefits of Monitoring SQL Server for Performance
12.2 Monitoring in a Dynamic Environment
12.3 Monitoring with SQL Profiler
12.4 Uses of SQL Profiler
12.5 Starting SQL Profiler
12.5.1 SQL Profiler Terminology
12.5.2 SQL Profiler Scenarios
12.5.3 Monitoring with SQL Profiler Event Categories
12.6 Monitoring with System Monitor
12.7 Monitoring with SQL Server Enterprise Manager
12.8 Monitoring the Error Logs
12.8.1 Comparing Error and Application Log Output
12.8.2 Viewing the SQL Server Error Log: SQL Server 2000
12.8.3 Viewing the Windows Application Log
12.9 Monitoring with Transact – SQL Statements
12.10 Monitoring with SNMP
12.10.1 SNMP Terminology
12.11 Enabling SQL Server Support of SNMP on Windows 98
12.12 Enabling SQL Server MIB
12.13 Copying the MSSQL – MIB to an SNMP Workstation
12.14 Summary
12.15 Keywords
12.16 Review Questions
12.17 Further Readings

Objectives

After studying this unit, you will be able to:

- Know about different tools and techniques available for monitoring performance
- Learn how to do monitoring with SQL Profiler
- Understand concept of monitoring with system monitor
- Analyze monitoring with the help of the Error Logs

Notes

Introduction

The goal of monitoring databases is to assess how a server is performing. Effective monitoring involves taking periodic snapshots of current performance to isolate processes that are causing problems, and gathering data continuously over time to track performance trends.

Ongoing evaluation of the database performance helps you minimize response times and maximize throughput, yielding optimal performance. Efficient network traffic, disk I/O, and CPU usage are key to peak performance. You need to thoroughly analyze the application requirements, understand the logical and physical structure of the data, assess database usage, and negotiate tradeoffs between conflicting uses such as online transaction processing (OLTP) versus decision support

12.1 Benefits of Monitoring SQL Server for Performance

Microsoft SQL Server and the Microsoft Windows operating system provide utilities that allow you to view the current condition of the database and to track performance as conditions change. There are a variety of tools and techniques that can be used to monitor Microsoft SQL Server.

Understanding how to monitor SQL Server can help you:

- Determine whether you can improve performance. For example, by monitoring the response times for frequently used queries, you can determine whether changes to the query or indexes on the tables are required.
- Evaluate user activity. For example, by monitoring users trying to connect to an instance of SQL Server, you can determine whether security is set up adequately and test applications or development systems. For example, by monitoring SQL queries as they are executed, you can determine whether they are written correctly and producing the expected results.
- Troubleshoot any problems or debug application components, such as stored procedures.

12.2 Monitoring in a Dynamic Environment

Monitoring is important because SQL Server provides a service in a dynamic environment. Changing conditions result in changing performance. In your evaluations, you can see performance changes as the number of users' increases, user access and connection methods change, database contents grow, client applications change, data in the applications changes, queries become more complex, and network traffic rises. By using SQL Server tools to monitor performance, you can associate some changes in performance with changing conditions and complex queries. The following scenarios provide examples:

- By monitoring the response times for frequently used queries, you can determine whether changes to the query or indexes on the tables where the queries execute are required.
- By monitoring Transact-SQL queries as they are executed, you can determine whether the queries are written correctly and producing the expected results.
- By monitoring users that try to connect to an instance of SQL Server, you can determine whether security is set up adequately and test applications or development systems.

Response time is the length of time required for the first row of the result set to be returned to the user in the form of visual confirmation that a query is being processed. Throughput is the total number of queries handled by the server during a specified period of time.

Notes

Notes As the number of users increases, so does the competition for a server's resources, which in turn increases response time and decreases overall.

12.3 Monitoring with SQL Profiler

SQL Profiler is a graphical tool that allows system administrators to monitor events in an instance of Microsoft® SQL Server™. You can capture and save data about each event to a file or SQL Server table to analyze later. For example, you can monitor a production environment to see which stored procedures are hampering performance by executing too slowly.

Use SQL Profiler to monitor only the events in which you are interested. If traces are becoming too large, you can filter them based on the information you want, so that only a subset of the event data is collected. Monitoring too many events adds overhead to the server and the monitoring process and can cause the trace file or trace table to grow very large, especially when the monitoring process takes place over a long period of time.

After you have traced events, SQL Profiler allows captured event data to be replayed against an instance of SQL Server, thereby effectively reexecuting the saved events as they occurred originally.

12.4 Uses of SQL Profiler

Use SQL Profiler to:

- Monitor the performance of an instance of SQL Server.
- Debug Transact-SQL statements and stored procedures.
- Identify slow-executing queries.
- Test SQL statements and stored procedures in the development phase of a project by single-stepping through statements to confirm that the code works as expected.
- Troubleshoot problems in SQL Server by capturing events on a production system and replaying them on a test system. This is useful for testing or debugging purposes and allows users to continue using the production system without interference.
- Audit and review activity that occurred on an instance of SQL Server. This allows a security administrator to review any of the auditing events, including the success and failure of a login attempt and the success and failure of permissions in accessing statements and objects.

SQL Profiler provides a graphical user interface to a set of stored procedures that can be used to monitor an instance of SQL Server. For example, it is possible to create your own application that uses SQL Profiler stored procedures to monitor SQL Server.

You must have at least 10 megabytes (MB) of free space to run SQL Profiler. If free space drops below 10 MB while you are using SQL Profiler, all SQL Profiler functions will stop.

12.5 Starting SQL Profiler

SQL Profiler is started from the Microsoft® Windows NT® 4.0, Microsoft Windows® 2000 or Microsoft Windows 98 **Start** menu, or from SQL Server Enterprise Manager.

Notes

With Windows Authentication mode, the user account that runs SQL Profiler must be granted permission to connect to an instance of SQL Server.

12.5.1 SQL Profiler Terminology

To use SQL Profiler, you need to understand the terminology that describes the way the tool functions. For example, you create a template that defines the data you want to collect. You collect this data by running a trace on the events defined in the template. While the trace is running, the event classes and data columns that describe the event data are displayed in SQL Profiler.

Template: A template defines the criteria for each event you want to monitor with SQL Profiler. For example, you can create a template, specifying which events, data columns, and filters to use. Then you can save the template and launch a trace with the current template settings. The trace data captured is based upon the options specified in the template. A template is not executed, and must be saved to a file with the .tdf extension.

Trace: A trace captures data based upon the selected events, data columns, and filters. For example, you can create a template to monitor exception errors. To do this, you would select to trace the **Exception** event class, and the **Error**, **State**, and **Severity** data columns, which need to be collected for the trace results to provide meaningful data. After you save the template, you can then run it as a trace, and collect data on any **Exception** events that occur in the server. This trace data can be saved and then replayed at a later date, or used immediately for analysis.

Filter: When you create a trace or template, you can define criteria to filter the data collected by the event. If traces are becoming too large, you can filter them based on the information you want, so that only a subset of the event data is collected. If a filter is not set, all events of the selected event classes are returned in the trace output. For example, you can limit the Microsoft® Windows® 2000 user names in the trace to specific users, reducing the output data to only those users in which you are interested.

Event Category: An event category defines the way events are grouped. For example, all lock events classes are grouped within the Locks event category. However, event categories only exist within SQL Profiler. This term does not reflect the way engine events are grouped.

Event: An event is an action generated within the Microsoft SQL Server™ engine. For example:

- The login connections, failures, and disconnections.
- The Transact-SQL SELECT, INSERT, UPDATE, and DELETE statements.
- The remote procedure call (RPC) batch status.
- The start or end of a stored procedure.
- The start or end of statements within stored procedures.
- The start or end of an SQL batch.
- An error written to the SQL Server error log.
- A lock acquired or released on a database object.
- An opened cursor.
- Security permissions checks.

All of the data that is generated as a result of an event is displayed in the trace in a single row. This row contains columns of data called event classes that describe the event in detail.

Notes

Event Class: An event class is the column that describes the event that was produced by the server. The event class determines the type of data collected, and not all data columns are applicable to all event classes. Examples of event classes include:

- **SQL:BatchCompleted**, which indicates the completion of an SQL batch.
- **Audit Login**, which collects all new connection events since the trace was started.
- **Audit Logout**, which collects all new disconnect events since the trace was started.
- **Lock:Acquired**, which indicates a lock on a resource, such as a data page, has been achieved.
- **Lock:Released**, which indicates a lock on a resource, such as a page, has been released.

Data Column: The data columns describe the data collected for each of the event classes captured in the trace. Because the event class determines the type of data collected, not all data columns are applicable to all event classes. For example, the **Binary Data** data column, when captured for the **Lock:Acquired** event class, contains the value of the locked page ID or row but has no value for the **Integer Data** data column. Default data columns are populated automatically for all event classes.

12.5.2 SQL Profiler Scenarios

Typically, you use SQL Profiler to:

- *Find the worst-performing queries:* For example, you can create a trace that captures events relating to TSQL and Stored Procedure event classes, specifically RPC:Completed and SQL:BatchCompleted. Include all data columns in the trace, group by Duration, and specify event criteria. For example, if you specify that the Duration of the event must be at least 1,000 milliseconds, you can eliminate short-running events from the trace. The Duration minimum value can be increased as required. If you want to monitor only one database at a time, specify a value for the Database ID event criteria.
- *Identify the cause of a deadlock:* For example, you can create a trace that captures events relating to TSQL and Stored Procedure event classes (RPC:Starting and SQL:BatchStarting) and Locks event classes (Lock:Deadlock and Lock:Deadlock Chain). Include all data columns in the trace and group by Event Class. If you want to monitor only one database at a time, specify a value for the Database ID event criteria.

  To view the connections involved in a deadlock, do one of the following:

  - Open the trace containing the captured data, group the data by ClientProcessID, and expand both connections involved in the deadlock.
  - Save the captured data to a trace file and open the trace file twice to make the file visible in two separate SQL Profiler windows. Group the captured data by ClientProcessID and then expand the client process ID involved in the deadlock; each deadlocked connection is in a separate window. Tile the windows to view the events causing the deadlock.
- *Monitor stored procedure performance:* For example, you can create a trace that captures events relating to Stored Procedures event classes (SP:Completed, SP:Starting, SP:StmtCompleted and SP:StmtStarting), and TSQL event classes (SQL:BatchStarting and SQL:BatchCompleted). Include all data columns in the trace and group by ClientProcessID. If you want to monitor only one database at a time, specify a value for the Database ID event criteria. Similarly, if you want to monitor only one stored procedure at a time, specify a value for the Object ID event criteria.
- *Audit Microsoft® SQL Server™ activity:* You can audit activity in SQL Server using SQL Profiler. For example, if the security administrator always needs to know who is logged in to the server, you can create a SQL Profiler trace that provides a complete view of users

Notes

who have logged in or out of the server. This information can then be used for legal purposes to document activity and for technical purposes to track security policy violations.

To set up a SQL Profiler trace that tracks users who have logged in or out of the server, do the following:

1. Create a trace, selecting **Audit Login Event**.
2. To return the appropriate information, specify the following data columns:
   i. **EventClass** (selected by default)
   ii. **EventSubClass**
   iii. **LoginSID**
   iv. **LoginName**
   v. Monitor Transact-SQL activity per user.

You can create a trace that captures events relating to the **Sessions** event class, **ExistingConnection**, and TSQL event classes. Include all data columns in the trace, do not specify any event criteria, and group the captured events by **DBUserName**.

Task Create a SQL Profiler trace that tracks users who have logged in or out of the server.

12.5.3 Monitoring with SQL Profiler Event Categories

In SQL Profiler, use event categories to monitor events in Microsoft® SQL Server™. Event categories contain event classes that have been grouped together within the SQL Profiler user interface.

The following table describes the SQL Profiler event categories and their associated event classes.

Event category	Description
Cursors	Collection of event classes produced by cursor operations.
Database	Collection of event classes produced when data or log files grow or shrink automatically.
Errors and Warnings	Collection of event classes produced when a SQL Server error or warning occurs (for example, an error during the compilation of a stored procedure or an exception in SQL Server).
Locks	Collection of event classes produced when a lock is acquired, cancelled, released, etc.
Objects	Collection of event classes produced when database objects are created, opened, closed, dropped, or deleted.
Performance	Collection of event classes produced when SQL data manipulation (DML) operators execute.
Scans	Collection tables and indexes are scanned.
Security Audit	Collection of event classes used to audit server activity.
Sessions	Collection of event classes produced by clients connecting to and disconnecting from an instance of SQL Server.
Stored Procedures	Collection of event classes produced by the execution of stored procedures.
Transactions	Collection of event classes produced by the execution of Microsoft Distributed Transaction Coordinator (MS DTC) transactions or by writing to the transaction log.
TSQL	Collection of event classes produced by the execution of Transact-SQL statements passed to an instance of SQL Server from the client.
User Configurable	Collection of user-configurable event classes.

Self Assessment Questions

Notes

Fill in the blanks:

1. SQL Profiler provides a ……...…….. interface to a set of stored procedures that can be used to monitor an instance of SQL Server.
2. The data columns describe the data collected for each of the ……...…….. classes captured in the trace.
3. An event class is the ……...…….. that describes the event that was produced by the server.
4. An event ……...…….. defines the way events are grouped.
5. When you create a trace or template, you can define criteria to ……...…….. the data collected by the event.
6. A ……...…….. captures data based upon the selected events, data columns, and filters.
7. A ……...…….. defines the criteria for each event you want to monitor with SQL Profiler

12.6 Monitoring with System Monitor

If you are running the Microsoft® Windows® 2000 operating system, use System Monitor (Performance Monitor in Microsoft Windows NT® 4.0) to measure the performance of Microsoft SQL Server™. You can view SQL Server objects and performance counters as well as the behavior of other objects, such as processors, memory, cache, threads, and processes. Each of these objects has an associated set of counters that measure device usage, queue lengths, delays, and other indicators of throughput and internal congestion.

System Monitor makes it possible to obtain up-to-the-second SQL Server activity and performance statistics. With this graphical tool, you can:

- View data simultaneously from any number of computers.
- View and change charts to reflect current activity, and show counter values that are updated at a user-defined frequency.
- Export data from charts, logs, alert logs, and reports to spreadsheet or database applications for further manipulation and printing.
- Add system alerts that list an event in the alert log and can notify you by reverting to the **Alert** view or issuing a network alert.
- Run a predefined application the first time or every time a counter value goes over or under a user-defined value.
- Create log files that contain data about various objects from different computers.
- Append to one file selected sections from other existing log files to form a long-term archive.
- View current-activity reports, or create reports from existing log files.
- Save individual chart, alert, log, or report settings, or the entire workspace setup for reuse when needed.

Notes You can use either the System Monitor or Performance Monitor to do these tasks.

Notes

12.7 Monitoring with SQL Server Enterprise Manager

Use SQL Server Enterprise Manager to view the following information about current Microsoft® SQL Server™ activity:

- Current user connections and locks.
- Process number, status, locks, and commands that active users are running.
- Objects that are locked, and the kinds of locks that are present.

If you are a system administrator, you can view additional information about a selected process, send a message to a user who is connected currently to an instance of SQL Server, or terminate a selected process.

Use the current activity window in SQL Server Enterprise Manager to perform ad hoc monitoring of an instance of SQL Server. This allows you to determine, at a glance, the volume and general types of activity on the system, for example:

- Current blocked and blocking transactions.
- Currently connected users on an instance of SQL Server and the last statement executed.
- Locks that are in effect.

SQL Server activity can be monitored using the **sp_who** and **sp_lock** system stored procedures.

Here are icons and descriptions of the icons in the current activity window.

Icon	Description
	Current Activity gives process and lock information at a designated time. This information is a snapshot taken every time you open or refresh **Current Activity**. The time of the snapshot is displayed in the left pane. **Current Activity** provides information about the processes (connections) running, the locks a certain connection is holding or trying to acquire, and the current and waiting locks on databases and tables.
	Process Info provides information about the current connections and activity in a system. A connection can be in three states: running, sleeping, or background. The database context is also displayed. There are some server processes, which are started before the **master** database is brought online, that have no database context.
	Running process that is waiting for a lock or user input.
	Sleeping process that is waiting for a lock or user input.
	Background process that wakes up periodically to execute work. SPID 2 (Lock Monitor), 3 (Lazy Writer) and 6 are background processes.
	Process (SPID) that is blocking one or more connections.
	Process (SPID) that is blocked by another connection.
	Process that is not blocking or being blocked.
	Process that is not blocking or being blocked.
	Table lock. If an index is involved, the index name is listed in the index column. The resource locator of the locked part is displayed in the resource column.
	Database lock.

Notes

Here are descriptions of the process information in the Current Activity window.

Item	Description
Process ID	SQL Server Process ID.
Context ID	Execution context ID used to uniquely identify the subthreads operating on behalf of a single process.
User	ID of the user who executed the command.
Database	Database currently being used by the process.
Status	Status of the process (for example, running, sleeping, runnable, and background).
Open Transactions	Number of open transactions for the process.
Command	Command currently being executed.
Application	Name of the application program being used by the process.
Wait Time	Current wait time in milliseconds. When the process is not waiting, the wait time is zero.
Wait Type	Indicates the name of the last or current wait type.
Wait Resources	Textual representation of a lock resource.
CPU	Cumulative CPU time for the process. The entry is updated only for processes performed on behalf of Transact-SQL statements executed when SET STATISTICS TIME ON has been activated in the same session. The CPU column is updated when a query has been executed with SET STATISTICS TIME ON. When zero is returned, SET STATISTICS TIME is OFF.
Physical IO	Cumulative disk reads and writes for the process.
Memory Usage	Number of pages in the procedure cache that are currently allocated to this process. A negative number indicates that the process is freeing memory allocated by another process.
Login Time	Time at which a client process logged into the server. For system processes, the time at which SQL Server startup occurred is displayed.
Last Batch	Last time a client process executed a remote stored procedure call or an EXECUTE statement. For system processes, the time at which SQL Server startup occurred is displayed.
Host	Name of the workstation.
Network Library	Column in which the client's network library is stored. Every client process comes in on a network connection. Network connections have a network library associated with them that allows them to make the connection.
Network Address	Assigned unique identifier for the network interface card on each user's workstation. When the user logs in, this identifier is inserted in the **Network Address** column.
Blocked By	Process ID (SPID) of a blocking process.
Blocking	Process ID (SPID) of processes that are blocked.

Here are descriptions of the lock information in the Current Activity window.

Item	Type	Description
spid	spid	Server process ID of the current user process.
ecid	ecid	Execution context ID. Represents the ID of a given thread associated with a specific spid.
Lock type	RID	Row identifier. Used to lock a single row individually within a table.
	KEY	Key; a row lock within an index. Used to protect key ranges in serializable transactions.

Contd.....

Notes

	PAG	Data or index page.
	EXT	Contiguous group of eight data pages or index pages.
	TAB	Entire table, including all data and indexes.
	DB	Database.
Lock mode	Shared (S)	Used for operations that do not change or update data (read-only operations), such as a SELECT statement.
	Update (U)	Used on resources that can be updated. Prevents a common form of deadlock that occurs when multiple sessions are reading, locking, and then potentially updating resources later.
	Exclusive (X)	Used for data modification operations, such as UPDATE, INSERT, or DELETE. Ensures that multiple updates cannot be made to the same resource at the same time.
	Intent	Used to establish a lock hierarchy.
	Schema	Used when an operation dependent on the schema of a table is executing. There are two types of schema locks: schema stability (Sch-S) and schema modification (Sch-M).
	Bulk update (BU)	Used when bulk copying data into a table and the **TABLOCK** hint is specified.
	RangeS_S	Shared range, shared resource lock; serializable range scan.
	RangeS_U	Shared range, update resource lock; serializable update scan.
	RangeI_N	Insert range, null resource lock. Used to test ranges before inserting a new key into an index.
	RangeX_X	Exclusive range, exclusive resource lock. Used when updating a key in a range.
Status	GRANT	Lock was obtained.
	WAIT	Lock is blocked by another process.
	CNVT	Lock is being converted to another lock. A lock being converted to another lock is held in one mode but is waiting to acquire a stronger lock mode (for example, update to exclusive). When diagnosing blocking issues, a CNVT can be considered similar to a WAIT.
Owner	Owner	The lock owner: xact (transaction), sess (session), or curs (cursor).
Index	Index	The index associated with the resource. If the index is clustered, you see the table name instead.
Resource	RID	Row identifier of the locked row within the table. The row is identified by a fileid:page:rid combination, where rid is the row identifier on the page.
	KEY	Hexadecimal number used internally by SQL Server.
	PAG	Page number. The page is identified by a fileid:page combination, where fileid is the **fileid** in the **sysfiles** table, and page is the logical page number within that file.
	EXT	First page number in the extent being locked. The page is identified by a fileid:page combination.
	TAB	No information is provided because the **ObjId** column already contains the object ID of the table.
	DB	No information is provided because the **dbid** column already

Self Assessment

Notes

State true or false:

8. System Monitor makes it possible to obtain up-to-the-second SQL Server activity and performance statistics.
9. The current activity window in SQL Server Enterprise Manager cannot be used to perform ad hoc monitoring of an instance of SQL Server.
10. SQL Server activity can be monitored using the **sp_who** and **sp_lock** system stored procedures.
11. When the process is waiting, the wait time is zero.

12.8 Monitoring the Error Logs

Microsoft® SQL Server™ logs events (although only certain system events and user-defined events) to the SQL Server error log and the Microsoft Windows® application log. Use the information in the error log to troubleshoot problems related to SQL Server.

The Windows application logs provide an overall picture of events that occur on the Windows NT® 4.0 and Windows 2000 systems, as well as events in SQL Server and SQL Server Agent. Use Event Viewer to view the Windows application log and to filter the information. For example, you can filter events, such as information, warning, error, success audit, and failure audit.

Both logs automatically timestamp all recorded events.

12.8.1 Comparing Error and Application Log Output

You can use both the SQL Server error log and the Windows application log to identify the cause of problems. For example, while monitoring the SQL Server error log, you may detect a certain set of messages for which you do not know the cause. By comparing the dates and times for events between these logs, you can narrow the list of probable causes.

12.8.2 Viewing the SQL Server Error Log: SQL Server 2000

View the Microsoft® SQL Server™ error log to ensure that processes have completed successfully (for example, backup and restore operations, batch commands, or other scripts and processes). This can be helpful to detect any current or potential problem areas, including automatic recovery messages (particularly if an instance of SQL Server has been stopped and restarted), kernel messages, and so on.

View the SQL Server error log by using SQL Server Enterprise Manager or any text editor. By default, the error log is located at Program Files\Microsoft SQL Server\Mssql\Log\Errorlog.

A new error log is created each time an instance of SQL Server is started, although the **sp_cycle_errorlog** system stored procedure can be used to cycle the error log files without having to restart the instance of SQL Server. Typically, SQL Server retains backups of the previous six logs and gives the most recent log backup the extension .1, the second most recent the extension .2, and so on. The current error log has no extension.

12.8.3 Viewing the Windows Application Log

When Microsoft® SQL Server™ is configured to use the Microsoft Windows® application log, each SQL Server session writes new events to that log. Unlike the SQL Server error log, a new application log is not created each time you start an instance of SQL Server.

Notes

View and manage the Windows application log by using Event Viewer in Microsoft Windows NT® 4.0 or Windows 2000.

There are three logs that can be viewed with Event Viewer.

Windows log type	Description
System log	Records events logged by the Windows NT 4.0 or Windows 2000 system components. For example, the failure of a driver or other system component to load during startup is recorded in the system log.
Security log	Records security events, such as failed login attempts. This helps track changes to the security system and identify possible breaches to security. For example, attempts to log on to the system may be recorded in the security log, depending on the audit settings in the User Manager. Only members of the **sysadmin** fixed server role can view the security log.
Application log	Records events that are logged by applications. For example, a database application might record a file error in the application log.

12.9 Monitoring with Transact – SQL Statements

Microsoft® SQL Server™ provides several Transact-SQL statements and system stored procedures that allow ad hoc monitoring of an instance of SQL Server. Use these statements when you want to gather, at a glance, information about server performance and activity. For example:

- Current locks.
- Current user activity.
- Last command batch submitted by a user.
- Data space used by a table or database.
- Space used by a transaction log.
- Oldest active transaction (including replicated transactions) in the database.
- Performance information relating to I/O, memory, and network throughput.
- Procedure cache usage.
- General statistics about SQL Server activity and usage, such as the amount of time the CPU has been performing SQL Server operations or the amount of time SQL Server has spent performing I/O operations.

Most of this information can also be monitored using SQL Server Enterprise Manager, SQL-DMO, or System Monitor (Performance Monitor in Microsoft Windows NT® 4.0).

12.10 Monitoring with SNMP

Simple Network Management Protocol (SNMP) is an application protocol that offers network management services. Using SNMP, you can monitor an instance of Microsoft® SQL Server™ across different platforms (for example, Microsoft Windows NT® 4.0, Microsoft Windows® 98, and UNIX).

With SQL Server and the Microsoft SQL Server Management Information Base (MSSQL-MIB), you can use SNMP applications to:

- Monitor the status of SQL Server installations. SNMP can only be used to monitor the default instances of SQL Server.
- Monitor performance information.

Notes

- Access databases.
- View server and database configuration parameters.

12.10.1 SNMP Terminology

Simple Network Management Protocol (SNMP) terms are defined in the following table.

Term	Description
SNMP	An application that monitors the status and performance of Microsoft® SQL Server™ installations, explores defined databases, and views server and database configuration parameters.
SNMP agent	SQL Server SNMP extension agent (Sqlsnmp.dll). Server software that extends the functionality of the SNMP service. The SNMP agent processes requests for data and data objects that reside on the local server.

12.11 Enabling SQL Server Support of SNMP on Windows 98

You can monitor remote connections to computers running Microsoft® Windows® 98 if your network uses Simple Network Management Protocol (SNMP).

The database controlled by an SNMP agent is known as SNMP Management Information Base (MIB). The values contained in an SNMP MIB can be shared with the SNMP MIB of another application.

Microsoft SQL Server™ Management Information Base (MSSQL-MIB), stored in the Mssql.mib file, and the SQL Server SNMP extension agent (Sqlsnmp.dll) are copied to the system by SQL Server Setup and are enabled if SNMP is running at the time of installation. SNMP can be activated or deactivated at any time by selecting the **Enable SNMP** check box in the **SQL Server Network Utility** dialog box.

12.12 Enabling SQL Server MIB

The database controlled by a Simple Network Management Protocol (SNMP) agent is known as SNMP Management Information Base (MIB). The values contained in an SNMP MIB can be shared with the SNMP MIB of another application.

Microsoft® SQL Server™ Management Information Base (MSSQL-MIB), stored in the Mssql.mib file, and the SQL Server SNMP extension agent (Sqlsnmp.dll) are copied to the system by SQL Server Setup and are enabled if SNMP is running at the time of installation.

12.13 Copying the MSSQL – MIB to an SNMP Workstation

For SNMP applications to monitor the status of a SQL Server installation, a copy of MSSQL-MIB, stored in the Mssql.mib file, must be placed on the monitoring workstation and loaded into the SNMP application. MSSQL-MIB enables the SNMP application to access and monitor the SQL Server SNMP extension agent on an instance of SQL Server.

The MSSQL.MIB file is a text file that contains the definitions of objects available to SNMP workstations. The file consists of read-only variables for monitoring general performance counters, the status of SQL Server installation and databases, and limited discovery of configuration options and database files. MSSQL.MIB does not define any writable objects.

The following table describes the SNMP tables. These tables are SNMP tables, not SQL Server tables.

Notes

SNMP table	Description
MssqlSrvTable	Contains a description of the SQL Server installation. Has a single row for each installation of SQL Server version 6.5 or earlier or multiple rows for each instance of SQL Server version 7.0 or SQL Server 2000 running on the server.
MssqlSrvInfoTable	Contains general information about the active SQL Server process, including performance counters.
MssqlSrvConfigParamTable	Lists SQL Server configuration parameters.
MssqlSrvDeviceTable	Contains an entry for each SQL Server database file defined on the system.
MssqlDbTable	Lists defined SQL Server databases. Contains a single row for each database.
MssqlDbOptionTable	Lists database options set for each SQL Server database.

Task Define the following terms:

1. SNMP
2. SNMP agent

12.14 Summary

- Microsoft® SQL Server™ 2000 provides a variety of tools that can be used to monitor the performance of an instance of SQL Server and the user activity that occurs in databases. Monitoring allows you to determine whether your database application is working efficiently and as expected, even as your application, database, and environment change.
- Microsoft® SQL Server™ provides a comprehensive set of tools for monitoring events in SQL Server such as SQL profiler, system monitor, SQL server enterprise manager, etc.
- System Monitor makes it possible to obtain up-to-the-second SQL Server activity and performance statistics.
- The database controlled by an SNMP agent is known as SNMP Management Information Base (MIB). The values contained in an SNMP MIB can be shared with the SNMP MIB of another application.
- System Monitor makes it possible to obtain up-to-the-second SQL Server activity and performance statistics.
- Microsoft® SQL Server™ provides several Transact-SQL statements and system stored procedures that allow ad hoc monitoring of an instance of SQL Server. Use these statements when you want to gather, at a glance, information about server performance and activity.

12.15 Keywords

Data Columns: The data columns describe the data collected for each of the event classes captured in the trace.

Event Category: An event category defines the way events are grouped.

Event Class: An event class is the column that describes the event that was produced by the server.

Notes

SNMP MIB: The database controlled by a Simple Network Management Protocol (SNMP) agent is known as SNMP Management Information Base (MIB).

SQL Server Enterprise Manager: SQL Server Enterprise Manager is a monitoring tool to view the information about current Microsoft® SQL Server™ activity.

SQL Server™ logs: Microsoft® SQL Server™ logs events to the SQL Server error log and the Microsoft Windows® application log.

System Monitor: System Monitor makes it possible to obtain up-to-the-second SQL Server activity and performance statistics.

Template: A template defines the criteria for each event you want to monitor with SQL Profiler.

Trace: A trace captures data based upon the selected events, data columns, and filters.

12.16 Review Questions

1. What are the Benefits of monitoring SQL Server for Performance?
2. Explain Monitoring with the help of SQL Profiler.
3. What are the Uses of SQL PROFILER?
4. What do you mean by the term: Event Class? Give Examples of event classes.
5. Discuss common SQL Profiler Scenarios.
6. Describe the procedure of Monitoring with SQL Profiler Event Categories.
7. How will you Monitoring SQL server with System Monitor? Discuss.
8. Write short notes on the following:
   (a) Monitoring the Error Logs
   (b) SNMP
   (c) Monitoring with Transact-SQL Statements
9. Discuss the concept of monitoring in a dynamic environment with examples.
10. What are the various SQL profilers event categories and their associated event classes? Discuss.
11. Make distinction between table lock and database lock.
12. Illustrate the concept of copying the MSSQL-MIB to an SNMP Workstation.

Answers: Self Assessment

1.	graphical user	2.	event
3.	column	4.	category
5.	filter	6.	trace
7.	template	8.	true
9.	false	10.	true
11.	false		

Notes

12.17 Further Readings

Books

C.J. Date, *Introduction to Database Systems*, Pearson Education.

Elmasri Navrate, *Fundamentals of Database Systems*, Pearson Education.

Peter Rob & Carlos Coronel, *Database Systems Design, Implementation and Management*, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, *Database Management Systems*, Tata McGraw Hill, 3rd Edition.

Silberschatz, Korth, *Database System Concepts*, McGraw Hill, 5th Edition.

Online links

www.sql-tutorial.net/

www.w3schools.com/sql/

www.sqlservertutorials.com/

Notes

Unit 13: Monitoring Database Modifications

CONTENTS

Objectives

Introduction

13.1 Defining DAM

13.2 Market Drivers

13.3 Use Cases

13.4 Common Use Cases for DAM

13.5 Common DAM architectures

13.6 User Benefits

13.7 Summary

13.8 Keywords

13.9 Review Questions

13.10 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand the meaning of Database Activity Monitoring
- Learn about common use cases for DAM
- Know about common DAM architectures
- Analyze the various market forces driving DAM tool deployment

Introduction

Continuous monitoring of the system, network, database operations, application, and other system components, ensures early detection of problems. Early detection improves the user's system experience because problems can be resolved faster. In addition, monitoring captures system metrics to indicate trends in system performance growth and recurring problems. This information can facilitate prevention, enforce security policies, and manage job processing. For the database server, a sound monitoring system must measure availability and detect events that can cause the database server to become unavailable, and provide immediate notification to responsible parties for critical failures.

The monitoring system itself must be highly available and adhere to the same operational best practices and availability practices as the resources it monitors. Failure of the monitoring system leaves all monitored systems unable to capture diagnostic data or alert the administrator of problems.

13.1 Defining DAM

Database Activity Monitors capture and record, at a minimum, all Structured Query Language (SQL) activity in real time or near real time, including database administrator activity, across

Notes

multiple database platforms; and can generate alerts on policy violations. While a number of tools can monitor various level of database activity, Database Activity Monitors are distinguished by five features:

1. The ability to independently monitor and audit all database activity, including administrator activity and SELECT transactions. Tools can record all SQL transactions: DML, DDL, DCL, (and sometimes TCL) activity.
2. The ability to store this activity securely outside the database.
3. The ability to aggregate and correlate activity from multiple heterogeneous Database Management Systems (DBMSs). Tools can work with multiple DBMSs (e.g., Oracle, Microsoft, IBM) and normalize transactions from different DBMSs despite differences between SQL flavors.
4. The ability to enforce separation of duties on database administrators.

   Auditing must include monitoring of DBA activity, and solutions should prevent DBA manipulation or tampering with logs or recorded activity.
5. The ability to generate alerts on policy violations. Tools don't just record activity, they provide real-time monitoring and rule-based alerting. For example, you might create a rule that generates an alert every time a DBA performs a select query on a credit card column which returns more than 5 results.

   Other tools provide some level of database monitoring, including Security Information and Event Management (SIEM), log management, and database management, but DAM products are distinguished by their ability to capture and parse all SQL in real time or near real time and monitor DBA activity.

Depending on the underlying platform, a key benefit of most DAM tools is the ability to perform this auditing without relying on local database logging, which often entails a substantial performance cost. All the major tools also offer other features beyond simple monitoring and alerting, ranging from vulnerability assessment to change management.

13.2 Market Drivers

DAM tools are extremely flexible and often deployed for what may appear to be totally unrelated reasons. Deployments are typically prompted by one of three drivers:

- ***Auditing for compliance:*** One of the biggest boosts to the DAM market has been increasing auditor requirements to record database activity for SOX (Sarbanes-Oxley) compliance. Some enterprises are required to record all database activity for SOX, and DAM tools can do this with less overhead than alternatives.
- ***As a compensating control for compliance:*** We are seeing greater use of DAM tools to address specific compliance requirements, even though database auditing itself isn't the specified control. The most common example is using DAM as an alternative to encrypting credit card numbers for PCI compliance.
- ***As a security control:*** DAM tools offer significant security benefits and can sometimes even be deployed in a blocking mode. They are particularly helpful in detecting and preventing data breaches for web facing databases and applications, or to protect sensitive internal databases through detection of unusual activity.

DAM tools are also beginning to expand into other areas of database and application security, as we'll see a bit later.

Notes

Notes Today, SOX compliance is the single biggest market driver, followed by PCI. Despite impressive capabilities, internally driven security initiatives motivate a distant third of DAM deployments as most database security projects seem to be driven by compliance needs.

13.3 Use Cases

Since Database Activity Monitoring is so versatile, here are a few examples of how it can be used:

- To enforce separation of duties on database administrators for SOX compliance by monitoring all their activity and generating SOX-specific reports for audits.
- If an application typically queries a database for credit card numbers, a DAM tool can generate an alert if the application requests more card numbers than a defined threshold. This can indicate that the application has been compromised via SQL injection or some other attack.
- To ensure that a service account only accesses a database from a defined source IP, and only runs a narrow group of authorized queries. This can alert on compromise of a service account either from the system that normally uses it, or if the account credentials show up in a connection from an unexpected system.
- For PCI compliance some organizations encrypt the database files or media where they're stored, and also use DAM to audit and alert on access to the credit card field. The encryption protects against physical theft, while the DAM protects against insider abuse and certain forms of external attack.
- As a change and configuration management tool. Some DAM tools offer closed-loop integration with external change management tools to track approved database changes implemented in SQL. Other tools can then track administrator activity and provide change management reports for manual reconciliation.

13.4 Common Use Cases for DAM

Privileged User Monitoring: Monitoring privileged users (or superusers), such as database administrators (DBAs), systems administrators (or sysadmins), developers, help desk, and outsourced personnel – who typically have unfettered access to corporate databases – is essential for protecting against both external and internal threats. Privileged user monitoring includes auditing all activities and transactions; identifying anomalous activities (such as viewing sensitive data, or creating new accounts with superuser privileges); and reconciling observed activities (such as adding or deleting tables) with authorized change requests.

Since most organizations are already protected at the perimeter level, indeed a major concern lies with the need to monitor and protect from privileged users. There is a high correlation therefore between Database Security and the need to protect from the insider threat. This is a complex task as most privileged users are capable of using sophisticated techniques to attack the database - stored procedures, triggers, views and obfuscated traffic - attacks that may be difficult to detect using traditional methods.

In addition, since targeted attacks frequently result in attackers gaining privileged user credentials, monitoring of privileged activities is also an effective way to identify compromised systems.

Notes

As a result, auditors are now demanding monitoring of privileged users for security best practices as well as a wide range of regulations. Privileged user monitoring helps ensure:

- Data privacy, so that only authorized applications and users are viewing sensitive data.
- Data governance, so that critical database structures and values are not being changed outside of corporate change control procedures.

Application Activity Monitoring: The primary purpose of application activity monitoring is to provide a greater level of end-user accountability and detect fraud (and other abuses of legitimate access) that occurs via enterprise applications, rather than via direct access to the database.

Multi-tier enterprise applications such as Oracle EBS, PeopleSoft, JD Edwards, SAP, Siebel Systems, Business Intelligence, and custom applications built on standard middle-tier servers such as IBM WebSphere and Oracle WebLogic Server mask the identity of end-users at the database transaction level. This is done with an optimization mechanism known as "connection pooling." Using pooled connections, the application aggregates all user traffic within a few database connections that are identified only by a generic service account name. Application activity monitoring allows organizations to associate specific database transactions with particular application end-users, in order to identify unauthorized or suspicious activities.

End-user accountability is often required for data governance requirements such as the Sarbanes–Oxley Act. New auditor guidance from the Public Company Accounting Oversight Board for SOX compliance has also increased the emphasis on anti-fraud controls.

Cyberattack Protection: SQL injection is a type of attack used to exploit bad coding practices in applications that use relational databases. The attacker uses the application to send a SQL statement that is composed from an application statement concatenated with an additional statement that the attacker introduces.

Many application developers compose SQL statements by concatenating strings and do not use prepared statement; in this case the application is susceptible to a SQL injection attack. The technique transforms an application SQL statement from an innocent SQL call to a malicious call that can cause unauthorized access, deletion of data, or theft of information.

One way that DAM can prevent SQL injection is by monitoring the application activity, generating a baseline of "normal behavior", and identifying an attack based on a divergence from normal SQL structures and normal sequences. Alternative approaches monitor the memory of the database, where both the database execution plan and the context of the SQL statements are visible, and based on policy can provide granular protection at the object level.

Task List the various uses of DAM.

13.5 Common DAM Architectures

Interception-based: Most modern DAM systems collect what the database is doing by being able to "see" the communications between the database client and the database server. What DAM systems do is find places where they can view the communication stream and get the requests and responses without requiring participation from the database. The interception itself can be done at multiple points such as the database memory (e.g. the SGA), at the network (using a network TAP or a SPAN port if the communication is not encrypted), at the operating system level, or at the level of the database libraries.

If there is unencrypted network traffic, then packet sniffing can be used. The advantage is that no processing is done on the host, however the main disadvantage is that both local traffic and

Notes

sophisticated intra-database attacks will not be detected. To capture local access some network based vendors deploy a probe that runs on the host. This probe intercepts all local access and can also intercept all networked access in case you do not want to use network gear or in case the database communications are encrypted. However, since the agent does not do all the processing — instead it relays the data to the DAM appliance where all the processing occurs — it may impact network performance with all of the local traffic and real-time session termination may be too slow to interrupt unauthorized queries.

Memory-based: Some DAM systems have a light weight sensor that attaches to the protected databases and continuously polls the system global area (SGA) to collect SQL statements as they are being performed. A similar architecture was previously used by performance optimization products that also used the SGA and other shared data structures.

Did u know? In the latest versions of this technology a light weight sensor runs on the host and attaches to the process at the OS level to inspect private data structures.

The advantages of this approach are significant:

- Complete coverage of all database transactions — the sensor covers traffic coming from the network, from the host, as well as from back-doors (stored procedures, triggers, views)
- A solution that is agnostic to most IT infrastructure variables - no need to re-architect the network, to open span ports or to worry about key management if the network is encrypted, and this model can also be used to protect databases deployed in virtualized environments or in the cloud

Log-based: Some DAM systems analyze and extract the information from the transaction logs (e.g., the redo logs). These systems use the fact that much of the data is stored within the redo logs and they scrape these logs. Unfortunately, not all of the information that is required is in the redo logs. For example, SELECT statements are not and so these systems will augment the data that they gather from the redo logs with data that they collect from the native audit trails as shown in Figure 3. These systems are a hybrid between a true DAM system (that is fully independent from the DBMS) and a SIEM which relies on data generated by the database. These architectures usually imply more overhead on the database server.

13.6 User Benefits

The user benefits can be quantified as:

1. ***Monitoring***

   i. *Privileged users monitoring:* DBAs, root, system admins – which have access to access and alter data either via the application either by logging in at the system OS or local console. Their access has to be monitored in order to prevent privileged users from accessing data, making modifications to schema or table structure, or creating or modifying user accounts or permissions

   ii. *User activity monitoring:* In order to track the users and the applications that connect to the database. Beside fraud access, an important aspect is also to monitor and eventually prevent also malicious or unintended activity of the legitimate users.

   iii. If possible, also the user accounts have to be constantly monitored in order to detect the dormant user accounts, and take appropriate action.

Notes

2. ***Risk and Compliance:*** the risk and security teams are seeking to implement tight controls around the data stores in order ensure data confidentiality and integrity while limiting access to privileged users and subsequently identifying fraudulent activities. The preventive security solutions and controls such as encryption and access management, are not effective for authorized/legitimate user access. Thus, the DAM solution can be successfully deployed in order to fulfill the security controls required by:

   i. Data Governance

   ii. Risk Management

   iii. Audit

   iv. Regulatory Compliance.

   The benefits provided by the DAM solution, via a tight integration with DLP and SIEM solutions respectively, allow the extension of the network controls and the security framework also to the databases and data stores.

3. ***Policy Enforcement:*** Database Firewall includes a complete set of predefined, customizable security and audit policies. Security alerts can be sent to SIEM, ticketing systems, and other third-party solutions to streamline business processes.

Business Impact

The solutions can be deployed offpath either inline. In order to comprehensively monitor and detect fraudulent activity, a solution must monitor all the "gateways" to the data store. Thus, the employment of software agents in order to monitor also the local activity (server console or other applications connecting to the database) is to be taken into consideration.

If the solutions are deployed offpath, then there is no impact on the monitored network segments.

If the solutions are to be deployed inline in protect mode (database firewall), then several considerations are to be taken into account:

- Usually the solution is deployed in transparent mode, with no IP addressing on the traffic interfaces. Thus, the DAM should have fail-open functionalities in order to allow the traffic to pass through in case of platform malfunction
- Minimum latency – the latency induced in the network has to be minimum
- The enforcement of the security policies is to be done in stages – first deployed non-intrusively, and upon successful testing only the security policies are to be enforced in place.
- Proper performance dimensioning in order to withstand peak traffic, in terms of both legitimate and malicious traffic; if the device cannot operate properly under heavy load this will have a direct impact on the business process
- Access policies have to be reviewed each time a modification has to be operated to the application, at both the application and at user access policies; failing to do this can result in blocking legitimate traffic and/or blocking legitimate users' access.
- Data audit – when turned on, it consumes very severely the resources – thus, a powerful enough hardware appliances has to be deployed.

Self Assessment

Notes

Fill in the blanks:

1. ..... must include monitoring of DBA activity, and solutions should prevent DBA manipulation or tampering with logs or recorded activity.
2. Most DAM tools is the ability to perform this auditing without relying on local .....
3. ..... are typically prompted by one of three drivers.
4. End-user accountability is often required for ..... requirements such as the Sarbanes–Oxley Act.
5. ..... is a type of attack used to exploit bad coding practices in applications that use relational databases.
6. If there is unencrypted network traffic, then ..... can be used.
7. ..... Is used in order to track the users and the applications that connect to the database.
8. ..... involves monitoring DBAs, root, system admins – which have access to access and alter data either via the application either by logging in at the system OS or local console.
9. The preventive security solutions and controls such as encryption and access management, are not effective for user access.
10. ..... includes a complete set of predefined, customizable security and audit policies.

13.7 Summary

- Database activity monitoring (DAM)/Database Firewall (DBF) monitors database activity to identify fraudulent, illegal or other undesirable behavior, by using embedded knowledge about database structures and access to analytics and reporting and enforce policies and control. The DAM/DBF solutions operates independently of the database management system (DBMS) audit functionality of the database itself. The DAM/DBF can be regarded to either as an alternative to the DBMS functionality (due to heavy overload on the database servers), either as complementary control to it.
- DAM solutions contain also database vulnerability assessment and user account audit, coupled with firewall file access monitoring and web application monitoring.
- Auditing must include monitoring of DBA activity, and solutions should prevent DBA manipulation or tampering with logs or recorded activity.
- Most DAM tools is the ability to perform this auditing without relying on local database logging
- Deployments are typically prompted by one of three drivers.
- SQL injection is a type of attack used to exploit bad coding practices in applications that use relational databases.
- If there is unencrypted network traffic, then packet sniffing can be used.
- User activity monitoring in order to track the users and the applications that connect to the database.

Notes

- Privileged users monitoring – DBAs, root, system admins – which have access to access and alter data either via the application either by logging in at the system OS or local console.
- The preventive security solutions and controls such as encryption and access management, are not effective for authorized/legitimate user access.
- Database Firewall includes a complete set of predefined, customizable security and audit policies

13.8 Keywords

Application Activity Monitoring: The primary purpose of application activity monitoring is to provide a greater level of end-user accountability and detect fraud (and other abuses of legitimate access) that occurs via enterprise applications, rather than via direct access to the database.

Database activity monitoring: Database activity monitoring (DAM) / Database Firewall (DBF) monitors database activity to identify fraudulent, illegal or other undesirable behavior, by using embedded knowledge about database structures and access to analytics and reporting and enforce policies and control.

Database Firewall: A tool that includes a complete set of predefined, customizable security and audit policies.

Privileged user Monitoring: This includes auditing all activities and transactions; identifying anomalous activities (such as viewing sensitive data, or creating new accounts with superuser privileges); and reconciling observed activities (such as adding or deleting tables) with authorized change requests.

SQL injection: It is a type of attack used to exploit bad coding practices in applications that use relational databases.

User Activity Monitoring: in order to track the users and the applications that connect to the database.

13.9 Review Questions

1. Discuss the need of database monitoring.
2. Describe the concepts of Database activity monitoring.
3. Explain the use of database monitoring.
4. Briefly describe the key features of database monitoring.
5. Explain the two models of DAM architecture.
6. Quantify the user benefits of database monitoring.
7. Explain the concept of Privileged User Monitoring.
8. Discuss the various market factors driving the deployment of DAM tool.

Answers: Self Assessment

1. Auditing
2. database logging
3. Deployments
4. data governance

Notes

5. SQL injection
6. packet sniffing
7. User activity monitoring
8. Privileged users monitoring
9. authorized/legitimate
10. Database Firewall

13.10 Further Readings

Books

C.J. Date, *Introduction to Database Systems*, Pearson Education.

Elmasri Navrate, *Fundamentals of Database Systems*, Pearson Education.

Peter Rob & Carlos Coronel, *Database Systems Design, Implementation and Management*, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, *Database Management Systems*, Tata McGraw Hill, 3rd Edition.

Silberschatz, Korth, *Database System Concepts*, McGraw Hill, 5th Edition.

Online links

www.sql-tutorial.net/

www.w3schools.com/sql/

www.sqlservertutorials.com/

Notes

Unit 14: SQL Server Integration Services

CONTENTS

Objectives

Introduction

14.1 Meaning of SSIS

14.2 The SSIS Import/Export Wizard

14.3 Process to Import and Export Data Into and From SQL Server

14.4 Features of the Data Flow Task

14.5 SSIS Components

14.6 Summary

14.7 Keywords

14.8 Review Questions

14.9 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand the meaning of SSIS
- Describe the features of SSIS
- Discuss how to create a SSIS Package
- Know about the components of a SSIS Package
- Learn how to use SQL Server Import and Export Wizard to create a data-transfer package.

Introduction

A very simple way to know about SSIS is that it's the solution for automating SQL Server. SSIS provides a way to build packages made up of tasks that can move data around from place to place and alter it on the way. There are visual designers (hosted within Business Intelligence Development Studio) to help you build these packages as well as an API for programming SSIS objects from other applications. In this unit, you'll learn the various features of SSIS, how to import and export data into and from SQL server as well as how to build and use SSIS packages.

14.1 Meaning of SSIS

SQL Server Integration Services (SSIS) is a platform for building high performance data integration and workflow solutions. It allows creation of packages or SSIS packages which are made up of tasks that can move data from source to destination and alter it if required.

SSIS is basically an ETL (Extraction, Transformation, and Load) tool whose main purpose is to do extraction, transformation and loading of data but it can be used for several other purposes for example, to automate maintenance of SQL Server databases, update multidimensional cube data etc., as well.

Notes

SSIS is a component of SQL Server 2005/2008 and is the successor of DTS (Data Transformation Services) which had been in SQL Server 7.0/2000. From an end-user perspective DTS and SSIS may appear similar, however they are actually quite different. SSIS has been completely written from scratch and overcomes several limitations of DTS. Though the list of differences between DTS and SSIS is quite large, something of note is that the internal architecture of SSIS is completely different from DTS. It has segregated the Data Flow Engine from the Control Flow Engine or SSIS Runtime Engine and hence improves the performance by a significant amount.

Notes In this unit, when referring to "SSIS 2008" it is the SSIS version that comes with SQL Server 2008 whereas "SSIS 2005" refers to the SSIS version that comes with SQL Server 2005.

Developers tasked with creating or maintaining SSIS packages use a visual development tool based on Microsoft Visual Studio called the SQL Server Business Intelligence Development Studio (BIDS). It allows users to edit SSIS packages using a drag-and-drop user interface. A scripting environment in which to write programming code is also available in the tool. A package holds a variety of elements that define a workflow. Upon package execution, the tool provides color-coded, real-time monitoring.

Connections

A connection includes the information necessary to connect to a particular data source. Tasks can reference the connection by its name, allowing the details of the connection to be changed or configured at run time.

Task A task is an atomic work unit that performs some action. There are a couple of dozen tasks that ship in the box, ranging from the file system task (which can copy or move files) to the data transformation task. The data transformation task actually copies data; it implements the ETL features of the product.

Precedence Constraints

Tasks are linked by precedence constraints. The precedence constraint preceding a particular task must be met before that task executes. The run time supports executing tasks in parallel if their precedence constraints so allow. Constraints may otherwise allow different paths of execution depending on the success or failure of other tasks. Together with the tasks, precedence constraints comprise the **workflow** of the package.

Event Handlers

A workflow can be designed for a number of events in the different scopes where they might occur. In this way, tasks may be executed in response to happenings within the package —such as cleaning up after errors.

Variables

Tasks may reference variables to store results, make decisions, or affect their configuration.

A package may be saved to a file or to a store with a hierarchical namespace within a SQL Server instance. In either case, the package content is persisted in XML.

Notes

Once completed, the designer also allows the user to start the package's execution. Once started, the package may be readily debugged or monitored.

14.2 The SSIS Import/Export Wizard

Though SSIS is almost infinitely customizable, Microsoft has produced a simple wizard to handle some of the most common ETL tasks: importing data to or exporting data from a SQL Server database. The Import and Export Wizard protects you from the complexity of SSIS while allowing you to move data between any of these data sources:

- SQL Server databases
- Flat files

Microsoft Access databases

- Microsoft Excel worksheets
- Other OLE DB providers

Did u know? You can launch the Import and Export wizard from the Tasks entry on the shortcut menu of any database in the Object Explorer window of SQL Server Management Studio.

The SSIS Import/Export Wizard lets the user create packages that move data from a single data source to a destination with no transformations. The Wizard can quickly move data from a variety of source types to a variety of destination types, including text files and other SQL Server instances.

Self Assessment

Fill in the blanks:

1. The package content is persisted in …….....
2. A task is an atomic work unit that performs some …….....
3. Together with the tasks, precedence constraints comprise the ……..... of the package.
4. A ……..... includes the information necessary to connect to a particular data source.
5. ETL stands for extraction, ……..... , and Load.

14.3 Process to Import and Export Data into and From SQL Server

To import some data using the Import and Export Wizard, follow these steps:

1. Launch SQL Server Management Studio and log in to your test server.
2. Open a new query window.
3. Select the master database from the Available Databases combo box on the toolbar.
4. Enter this text into the query window:

   CREATE DATABASE Chapter16
5. Click the Execute toolbar button to create a new database.
6. Expand the Databases node in Object Explorer
7. Right-click on the Chapter16 database and select Tasks- Import Data.

Notes

8. Read the first page of the Import and Export Wizard and click Next.
9. Select SQL Native Client for the data source and provide login information for your test server.
10. Select the AdventureWorks database as the source of the data to import.
11. Click Next.
12. Because you're importing data, the next page of the wizard will default to connection information for the Chapter16 database. Click Next.
13. Select Copy Data From One or More Tables or Views and click Next.

> *Notes* that if you only want to import part of a table you can use a query as the data source instead.

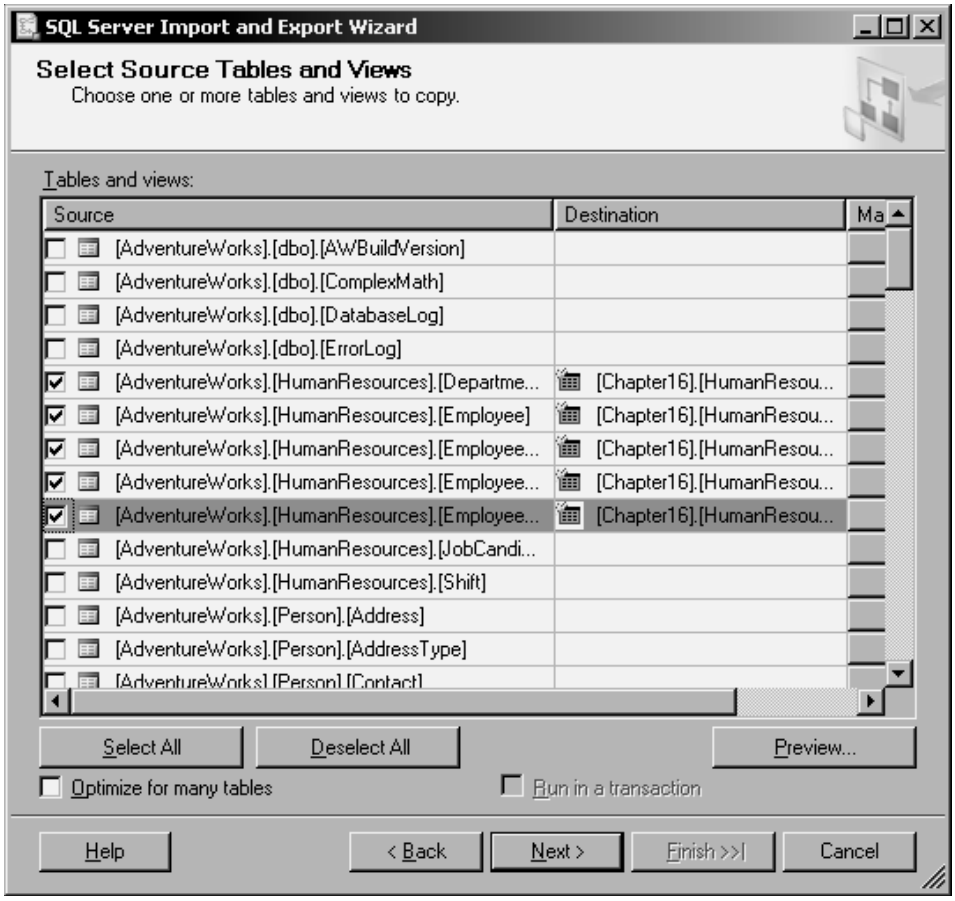
14. Select the HumanResources.Department, HumanResources.Employee,

    HumanResources.EmployeeAddress,

    HumanResources.EmployeeDepartmentHistory, and

    HumanResources.EmployeePayHistory tables, as show in Figure 14.1. As you select tables, the wizard will automatically assign names for the target tables.

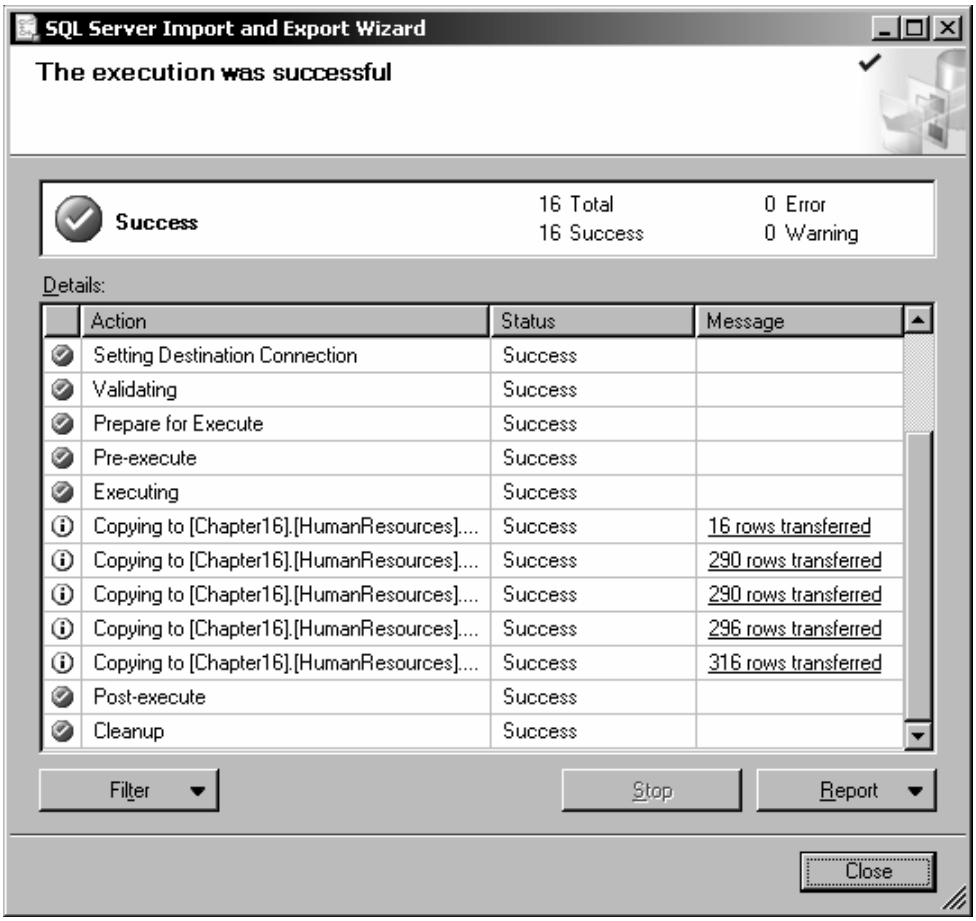
Figure 14.1



Notes

15. Click the Edit button in the Mapping column for the HumanResources.Department table.
16. The Column Mappings dialog box lets you change the name, data type, and other properties of the destination table columns. You can also set other options here, such as whether to overwrite or append data when importing data to an existing table. Click Cancel when you're done inspecting the options.
17. Click Next.
18. Check Execute Immediately and click Next.
19. Click Finish to perform the import. SQL Server will display progress as it performs the import, as shown in Figure 14.2.

Figure 14.2



20. Click Close to dismiss the report.
21. Expand the Tables node of the Chapter16 database to verify that the import succeeded.

Notes In addition to executing its operations immediately, the Import and Export Wizard can also save a package for later execution.

Notes

14.4 Features of the Data Flow Task

SSIS provides the following built-in transformations:

- Conditional Split,
- Multicast
- Union-All, Merge, and Merge Join
- Sort
- Fuzzy Grouping
- Lookup and Fuzzy Lookup
- Percentage Sampling and Row Sampling
- Copy/Map, Data Conversion, and Derived Column
- Aggregation
- Data Mining Model Training, Data Mining Query, Partition Processing, and Dimension Processing
- Pivot and Unpivot
- Slowly Changing Dimension
- Script Component
- Audit
- Cache Transform
- Export and Import Column
- OLE DB Command
- Row Count
- Term Extraction
- Term Lookup

The Conditional Split transformation is used to speed up the query on the source table based on a particular condition. It is similar to the "if..else" construct in the **C** language.

14.5 SSIS Components

As described in the introduction, SSIS creates packages which are composed of tasks that can move data from source to destination, and if necessary transform it. Within SSIS package the workflow can be defined, the SSIS runtime engine ensures the tasks inside the package are executed according to the workflow. Following is a description of the different tasks/components/executables of a package.

Package

A package is a collection of tasks which are executed in an orderly fashion by SSIS runtime engine. It is an XML file, which can be saved on SQL Server or on a file system. A package can be executed by SQL Server Agent Job, DTEXEC command (a command line utility bundled with SSIS to execute a package; another similar utility DTEXECUI, has a GUI), from BIDS environment or by calling one package by another package (achieves modular approach). You can use DTUTIL utility to move package from file system to SQL Server or vice versa. Alternatively the undocumented sp_dts_getpackage/sp_ssis_getpackage and sp_dts_putpackage/sp_ssis_putpackage stored procedures which reside in msdb system database can be used.

Notes

Control Flow

Handles the main workflow of the package and determines processing sequence within the package. It consists of containers, different kinds of work flow tasks and precedence constraints.

Control Flow Tasks

A task is an individual unit of work. SSIS provides several inbuilt control flow tasks which perform a variety of workflow actions. They provide functionality to the package in much the same way that a method does in programming language. All the inbuilt tasks are operational task except Data Flow Task. Though there are several dozen inbuilt tasks for use, if required they can be extended and custom tasks can be written using VB/C# etc.

Containers

Containers group a variety of package components (including other containers), affect their scope, sequence of execution and mutual interaction. They are used to create logical groups of tasks. There are four types of containers in SSIS listed below:

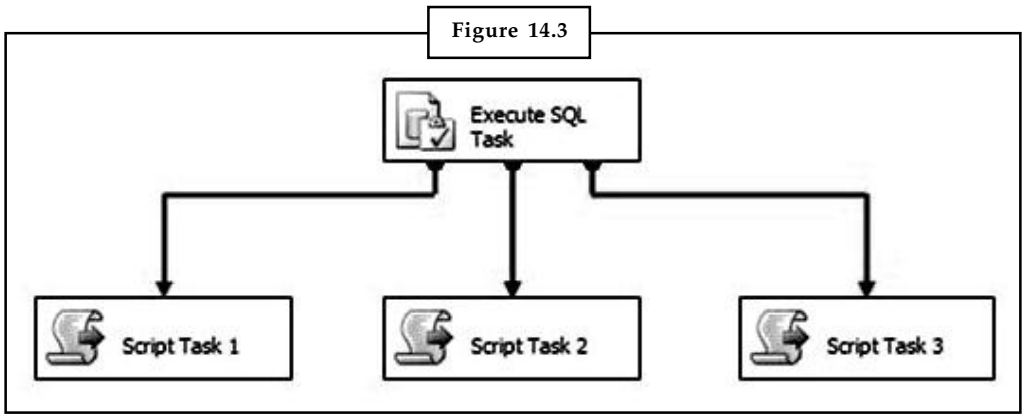
- Task Host Containers – Default container, every task falls into it.
- Sequence Containers – Defines a subset of the overall package control flow.
- For Loop Containers – Defines a repeating control flow in a package.
- For Each Loop Containers – Loops for collection, enumerates through a collection for example it will be used when each record of a record-set needs to be processed.

Precedence Constraints

Precedence constraints link the items in a package into a logical flow and specify the conditions upon which the items are executed. It provides an ordinal relationship between various items in the package; which helps manage the order the tasks will execute. It directs the order of task execution and defines links among containers and tasks; evaluates conditions that determine the sequence in which they are processed. More specifically, they provide transition from one task or container to another.

The condition can either be Constraint or Expression or both. The constraint can be Success (Green Line), Failure (Red Line) and Complete (Blue Line). The package in the image below shows Script Task 1 will be executed only if the Execute SQL Task completed successfully; Script Task 2 will be executed irrespective of whether the Execute SQL Task completed successfully or failed; Script Task 3 will be executed only if the Execute SQL Task failed.

Figure 14.3

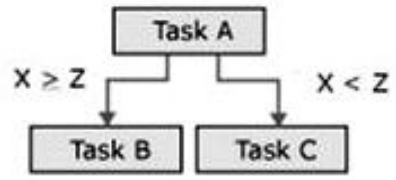


Notes

Apart from the above discussed constraints, conditions can also be defined as an expression with precedence constraints that is evaluated at runtime; depending on its value the transition is decided. In the image below, After Task A, Task B will be executed if the value of X >= Z or Task C will be executed if the value of X < Z. Constraint and expression can also be combined in a single condition with either AND or OR operator.

Task What are Precedence constraints? What is its role in a SSIS package?

Figure 14.4



Variables

The concept of a variable in SSIS is same as the variables in any other programming language. It provides temporary storage for parameters whose values can change from one package execution to another, accommodating package reusability. It is used to dynamically configure a package at runtime. For example, to execute the same T-SQL statement or a script against a different set of connections. Depending on the place where a variable has been defined, its scope varies. Variables can be declared at package, container, task or handlers level.

In SSIS, there are two types of variables – System (predefined) variables whose values are set by SSIS (ErrorCode, ErrorDescription, MachineName, PackageName, StartTime etc.) and cannot be changed; User variables, created as required at the time of package development, can be assigned a value of the corresponding type.

Note: An exception applies here, there is a system variable called "Propagate" whose value can be changed from its default value TRUE to FALSE to stop event bubbling from a task to its parent and grandparent. T

Connection Managers

A connection manager is a logical representation of a connection. SSIS provides different types of connection managers which use different data providers and enable packages to connect to a variety of data sources and servers.

A package can have multiple instances of connection managers and one connection manager can be used by multiple tasks in the package.

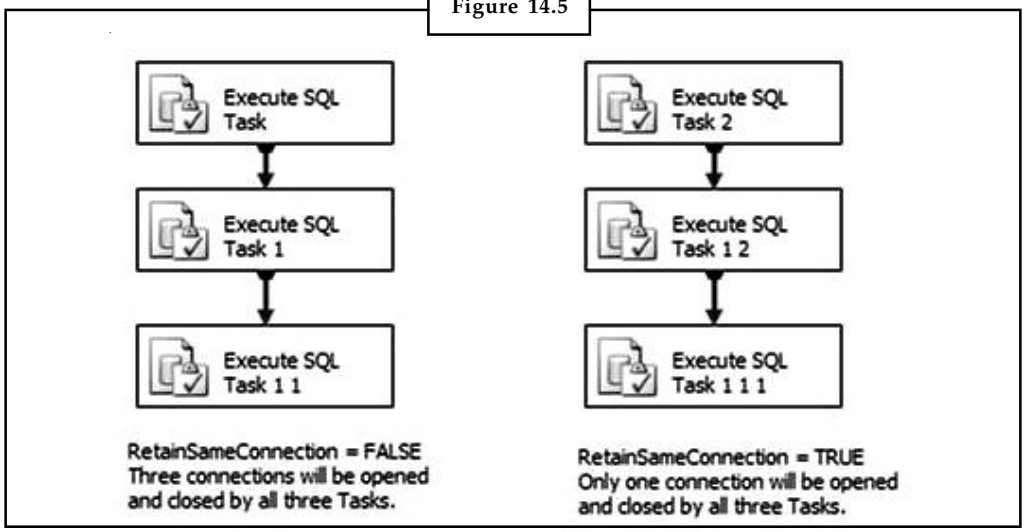
Some examples of connection managers are:

- ADO Connection Manager – Connects to ActiveX Data Objects (ADO) objects.
- ADO.NET Connection Manager – Connects to a data source by using a .NET provider.
- OLEDB Connection Manager – Connects to a data source by using an OLE DB provider.
- Flat File Connection Manager – Connect to data in a single flat file.
- FTP Connection Manager – Connect to an FTP server.

Notes

By default, every task that uses a connection manager during execution opens a connection, performs the operation and closes the connection before moving to the next task. Each task has its own connection. Consider a scenario where there are three tasks in package and they use the same connection manager; during runtime there would be three connections open and closed at the source. However, ideally all three tasks would be executed in a single connection; so that only one connection is open to the source irrespective of how many tasks use the connection. The RetainSameConnection property on the OLE DB Connection Manager enables multiple tasks to run in a single connection if the RetainSameConnection property equals TRUE.

Figure 14.5



Self Assessment

State true or false:

6. A package can have multiple instances of connection managers and one connection manager can be used by multiple tasks in the package.
7. In SSIS, there are three types of variables.
8. Precedence constraints link the items in a package into a logical flow and specify the conditions upon which the items are executed.
9. There are two types of containers in SSIS.
10. A package is a collection of tasks which are executed in an orderly fashion by SSIS runtime engine.

14.6 Summary

- SSIS provides a way to build packages made up of tasks that can move data around from place to place and alter it on the way.
- SSIS is basically an ETL (Extraction, Transformation, and Load) tool whose main purpose is to do extraction, transformation and loading of data but it can be used for several other purposes for example, to automate maintenance of SQL Server databases, update multidimensional cube data etc., as well.

Notes

- SSIS creates packages which are composed of tasks that can move data from source to destination, and if necessary transform it.
- Within SSIS package the workflow can be defined, the SSIS runtime engine ensures the tasks inside the package are executed according to the workflow.

14.7 Keywords

Connection Manager: A connection manager is a logical representation of a connection.

Connections: A connection includes the information necessary to connect to a particular data source.

Containers: Containers group a variety of package components (including other containers), affect their scope, sequence of execution and mutual interaction.

Event Handlers: A workflow can be designed for a number of events in the different scopes where they might occur.

Package: A package is a collection of tasks which are executed in an orderly fashion by SSIS runtime engine.

Precedence Constraints: Tasks are linked by precedence constraints. The precedence constraint preceding a particular task must be met before that task executes.

Tasks: A task is an atomic work unit that performs some action.

Variables: Tasks may reference variables to store results, make decisions, or affect their configuration.

14.8 Review Questions

1. Explain the concept of SSIS.
2. Briefly describe the process to import and export data into and from sql server.
3. Discuss the concept of "**The SSIS Import/Export Wizard**"
4. Describe the various components of SSIS.
5. What is a connection manager? Give few examples of connection managers.
6. Discuss the concept of Containers? What are the four types of containers in SSIS?
7. Write short notes on following:
   (a) Precedence constraints
   (b) Event handlers
   (c) Connection Manager
   (d) Variables
8. Discuss the various features of the data flow task.
9. Explain how to create SSIS package.

Answers: Self Assessment

1. XML
2. Action
3. Workflow
4. Connection

Notes

5.	Transformation	6.	True
7.	False	8.	True
9.	False	10.	True

14.9 Further Readings

Books

C.J. Date, *Introduction to Database Systems*, Pearson Education.

Elmasri Navrate, *Fundamentals of Database Systems*, Pearson Education.

Peter Rob & Carlos Coronel, *Database Systems Design, Implementation and Management*, 7th Edition.

Raghurama Krishnan, Johannes Gehrke, *Database Management Systems*, Tata McGraw Hill, 3rd Edition.

Silberschatz, Korth, *Database System Concepts*, McGraw Hill, 5th Edition.

Online links

www.sql-tutorial.net/

www.w3schools.com/sql/

www.sqlservertutorials.com/