MICROSOFT
TRAINING
AND CERTIFICATION

Microsoft Official Curriculum

Module 1: SQL Server Overview

Contents

Microsoft

Project Lead: Rich Rose
Instructional Designers: Rich Rose, Cheryl Hoople, Marilyn McGill
Instructional Software Design Engineers: Karl Dehmer, Carl Raebler, Rick Byham
Technical Lead: Karl Dehmer
Subject Matter Experts: Karl Dehmer, Carl Raebler, Rick Byham
Graphic Artist: Kirsten Larson (Independent Contractor)
Editing Manager: Lynette Skinner
Editor: Wendy Cleary
Copy Editor: Edward McKillop (S&T Consulting)
Production Manager: Miracle Davis
Production Coordinator: Jenny Boe
Production Support: Lori Walker (S&T Consulting)
Test Manager: Sid Benavente
Courseware Testing: TestingTesting123
Classroom Automation: Lorrin Smith-Bates
Creative Director, Media/Sim Services: David Mahlmann
Web Development Lead: Lisa Pease
CD Build Specialist: Julie Challenger
Online Support: David Myka (S&T Consulting)
Localization Manager: Rick Terek
Operations Coordinator: John Williams
Manufacturing Support: Laura King; Kathy Hershey
Lead Product Manager, Release Management: Bo Galford
Lead Product Manager, Data Base: Margo Crandall
Group Manager, Courseware Infrastructure: David Bramble
Group Product Manager, Content Development: Dean Murray
General Manager: Robert Stewart

Instructor Notes

Presentation:
60 Minutes

Lab:
30 Minutes

This module is a high-level overview of Microsoft® SQL Server™ 2000 platforms, architecture, components, and security. It also identifies and defines key SQL Server terminology and concepts. This module discusses how well SQL Server integrates with Microsoft Windows® 2000 and other Microsoft server applications. It concludes with an overview of SQL Server database administration and implementation activities, as well as SQL Server application design options.

After completing this module, students will be able to:

- Describe SQL Server 2000 and its supported operating system platforms.
- Describe SQL Server integration with Windows 2000 and other server applications.
- Describe SQL Server databases.
- Describe SQL Server security.
- Describe SQL Server administration and implementation activities, as well as SQL Server application design options.

Materials and Preparation

This section provides the materials and preparation tasks that you need to teach this module.

Required Materials

To teach this module, you need the following materials:

- Microsoft PowerPoint® file 2073A_01.ppt.
- The C:\Moc\2073A\Demo\D01_Ex.sql example file, which contains all of the example scripts from the module, unless otherwise noted in the module.

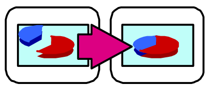
Preparation Tasks

To prepare for this module, you should:

- Read all of the materials for this module.
- Complete the lab.

Other Activities

This section provides procedures for implementing interactive activities to present or review information, such as games or role playing exercises.



Displaying the Animated PowerPoint Slides

All animated slides are identified with an icon of links on the lower-left corner of the slide.

► **To display the Client-Server Communication Process slide**

This slide shows how SQL Server processes a query.

1. Display the topic slide, where the client application submits a query. The client calls the database application programming interface (API) and passes the query. The database API uses a provider, driver, or DLL to encapsulate the query in one or more Tabular Data Stream (TDS) packets and pass them to the client Net-Library.
2. Advance to the first animation, where the client Net-Library packages the TDS packets into network protocol packets. The client Net-Library calls a Windows interprocess communication (IPC) API to send the network protocol packets to a server Net-Library by using the network protocol stack of the operating system. The appropriate server Net-Library extracts the TDS packets from the network protocol packets and passes the TDS packets to Open Data Services.
3. Advance to the next animation, where Open Data Services extracts the query from the TDS packets and passes the query to the relational engine. The relational engine then compiles the query into an optimized execution plan. It executes the execution plan. The relational engine communicates with the storage engine by using the OLE DB interface.
4. Advance to the next animation, where the storage engine transfers data from a database to data buffers and then passes rowsets containing data to the relational engine. The relational engine combines the rowsets into the final result set and passes the result set to Open Data Services.
5. Advance to the final animation, where Open Data Services packages the result set and returns it to the client application by using a server Net-Library, the network protocol stack, the client Net-Library, and the database API.

► To display the Database Objects slide

This slide illustrates SQL Server database objects to provide high-level definitions of each. Note that the information is based on a fictitious database.

1. Display the slide—the illustration of the **Employee** table appears initially.
2. Display the next image, in which the system-supplied and user-defined data types assigned to each column appear.
3. Display the next image, in which the three types of constraints appear. Briefly define PRIMARY KEY, FOREIGN KEY, and CHECK constraints. Also introduce the DEFAULT constraint (although it does not appear), and mention rules and defaults.
4. Display the next image, in which the clustered index on the **LastName** column appears. *Briefly* define the difference between clustered and nonclustered indexes. Other indexes that you would most likely create on this table include a nonclustered index on the **CtryCode** column with the FOREIGN KEY constraint, and a unique, nonclustered index on the **EmpNum** column (when PRIMARY KEY was defined).
5. Display the next image, in which the definition of the **EmployeePhoneView** view appears.
6. Display the next image, in which a square appears above the **Extension** column, indicating that a stored procedure could be written to update an employee's telephone extension.
7. Display the final image, in which a square appears above the **LastMod** column, indicating that an audit-trail trigger has been defined. The trigger automatically updates the value with the user name that last modified the row in the table.

Module Strategy

This module appears in two courses: course 2072A, *Administering a Microsoft SQL Server 2000 Database*, and course 2073A, *Programming a Microsoft SQL Server 2000 Database*. Survey students to determine their levels of knowledge, and then use that information to adjust the presentation of this module accordingly.

Use the following strategy to present this module:

- What Is SQL Server?

  Introduce SQL Server. Discuss the scalability of SQL Server and the role of a relational database management system (RDBMS).

  Point out that users manage two types of databases—online transaction processing (OLTP) databases, and online analytical processing (OLAP) databases. Emphasize that this course focuses on activities associated with OLTP databases and Transact-SQL.

  Point out that users do not access SQL Server and SQL Server 2000 Analysis Services directly, but that they use separate client applications that are written to access the data. Briefly review the client applications that access SQL Server.

  Describe the components involved in the client-server communication process and the role of each component.

  Emphasize that the client Net-Library must match one of the server Net-Libraries.

  Display the Client-Server Communication Process slide.

  Describe SQL Server services.

- SQL Server Integration

  Describe SQL Server and its supported operating systems. Point out that Internet browsers and third-party client applications running on various operating systems also can access SQL Server.

  Describe SQL Server integration with Windows 2000 and other Microsoft server applications. Discuss how SQL Server integrates with specific features of Windows 2000.

- SQL Server Databases

  Point out that an understanding of SQL Server database structure helps the user develop and implement a database effectively.

  Introduce the two types of SQL Server databases. Explain that system tables store information about the system or objects in a database.

  Briefly describe and point out the function of each database and then outline several typical designs for user databases. Point out that SQL Server supports multiple instances.

  Caution that deleting the **msdb** database eliminates most of the functionality of SQL Server Agent.

  Step through and narrate the animated slide that provides a high-level definition of SQL Server database objects. Note that the information in the illustration is fictitious and serves as a teaching example only.

Summarize and describe each database object, distinguishing between system-supplied and user-defined data types and between clustered and nonclustered indexes.

Refer students to SQL Server Books Online for a detailed discussion of metadata.

Emphasize that writing scripts that directly query the system tables is not recommended, and that students should use information schema views instead of system tables.

Point out that system functions return specific, single values. Also explain that information schema views in SQL Server conform to the ANSI SQL standard and generate multiple values.

- SQL Server Security

Point out that SQL Server validates users at two levels of security—login authentication and permissions validation on database user accounts and roles.

Describe how SQL Server authenticates login accounts. Point out that a user must have a login account to connect to SQL Server. Explain the difference between Windows 2000 authentication and SQL Server Authentication, and between Windows Authentication Mode and Mixed Mode.

Because students may find authentication modes and mechanisms confusing, point out that the server runs in an authentication mode and that a client connecting to a server running in Mixed Mode must choose an authentication mechanism.

Describe how SQL Server maps login authentication to user accounts and roles. Point out the difference between server-level and database-level roles and corresponding permissions.

Review how SQL Server validates permissions.

- Working with SQL Server

Discuss when to use scripts or a graphical tool to work with SQL Server. Provide examples.

Briefly identify common administrative tasks related to administering SQL Server.

Briefly demonstrate SQL Server Enterprise Manager, SQL Query Analyzer, and **osql**.

Describe each type of Help that SQL Server provides.

Briefly identify typical activities related to implementing a database.

Compare several physical architectures and logical software application designs that are available when planning a database. Point out that the slide illustrates only the most common architectural designs and deployment options.

Emphasize that in any design, the software application can physically reside on one or more servers. Briefly describe each application deployment option and go over the examples.

Describe the APIs to use when developing a relational database application for use with SQL Server. Refer students to SQL Server Books Online for a detailed discussion of programming interfaces.

Customization Information

This section identifies the lab setup requirements for a module and the configuration changes that occur on student computers during the labs. This information is provided to assist you in replicating or customizing Microsoft Official Curriculum (MOC) courseware.

Important The lab in this module is dependent on the classroom configuration that is specified in the Customization Information section at the end of the *Classroom Setup Guide* for course 2073A, *Programming a Microsoft SQL Server 2000 Database*.

Lab Setup

There are no lab setup requirements that affect replication or customization.

Lab Results

There are no configuration changes on student computers that affect replication or customization.

Overview

Topic Objective
To provide an overview of the module topics and objectives.

Lead-in
In this module, you will review SQL Server architecture, components, terminology, and concepts.

- **What Is SQL Server?**
- **SQL Server Integration**
- **SQL Server Databases**
- **SQL Server Security**
- **Working with SQL Server**

This module provides a high-level overview of Microsoft® SQL Server™ 2000 platforms, architecture, components, and security. It also identifies and defines key SQL Server terminology and concepts. This module discusses how SQL Server integrates with Microsoft Windows® 2000 and other Microsoft server applications. It concludes with an overview of SQL Server database administration and implementation activities, as well as SQL Server application design options.

After completing this module, you will be able to:

- Describe SQL Server 2000 and its supported operating system platforms.
- Describe SQL Server integration with Windows 2000 and other server applications.
- Describe SQL Server databases.
- Describe SQL Server security.
- Describe SQL Server administration and implementation activities, as well as SQL Server application design options.

◆ What Is SQL Server?

Topic Objective
To outline the topic.

Lead-in
You use SQL Server to manage two types of databases.

- **Introduction to SQL Server**
- **Client-Server Components**
- **Client-Server Communications**
- **SQL Server Services**

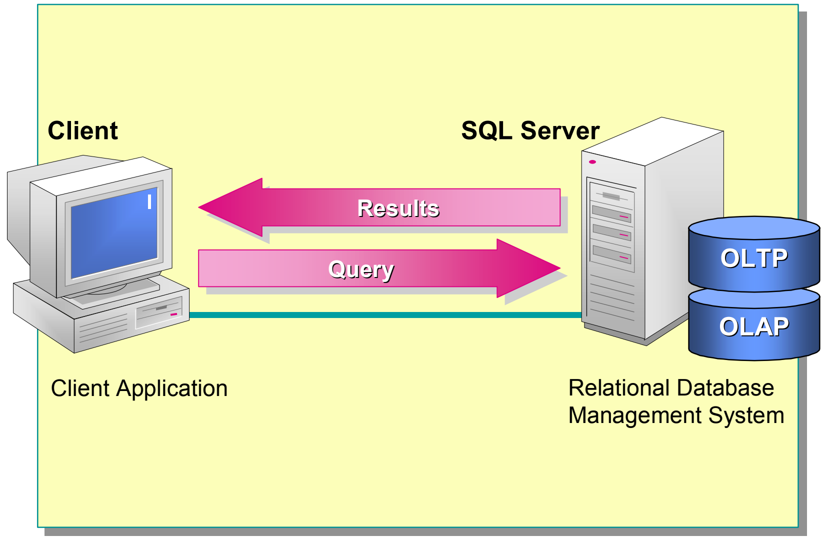
You use SQL Server to manage two types of databases—online transaction processing (OLTP) databases, and online analytical processing (OLAP) databases. Typically, separate clients access the databases by communicating over a network.

You can scale SQL Server up to terabyte-size databases and down to small business servers and portable computers. You can scale SQL Server out to multiple servers by using Windows Clustering in Windows 2000.

Introduction to SQL Server

Topic Objective
To introduce SQL Server.

Lead-in
You can use SQL Server to perform transaction processing, store and analyze data, and build new applications.



You can use SQL Server to perform transaction processing, store and analyze data, and build new applications.

SQL Server is a family of products and technologies that meets the data storage requirements of OLTP and OLAP environments. SQL Server is a relational database management system (RDBMS) that:

- Manages data storage for transactions and analysis.
- Responds to requests from client applications.
- Uses Transact-SQL, Extensible Markup Language (XML), multidimensional expressions (MDX), or SQL Distributed Management Objects (SQL-DMO) to send requests between a client and SQL Server.

Note This course focuses on activities associated with OLTP databases and Transact-SQL.

Delivery Tip
Emphasize that this course focuses on activities associated with OLTP databases and Transact-SQL.

Relational Database Management System

The RDBMS of SQL Server is responsible for:

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

Data Storage Models

SQL Server manages OLTP and OLAP databases.

OLTP Databases Data in an OLTP database is generally organized into relational tables to reduce redundant information and to increase the speed of updates. SQL Server enables a large number of users to perform transactions and simultaneously change real-time data in OLTP databases. Examples of OLTP databases include airline ticketing and banking transaction systems.

OLAP Databases OLAP technology organizes and summarizes large amounts of data so that an analyst can evaluate data quickly and in real time. SQL Server 2000 Analysis Services organizes this data to support a wide array of enterprise solutions, from corporate reporting and analysis to data modeling and decision support.

Client Applications

Users do not access SQL Server and Analysis Services directly; instead, they use separate client applications written to access the data. These applications access SQL Server by using:

Transact-SQL This query language, a version of Structured Query Language (SQL), is the primary database query and programming language that SQL Server uses.

XML This format returns data from queries and stored procedures by using URLs or templates over Hypertext Transfer Protocol (HTTP). You also can use XML to insert, delete, and update values in a database.

MDX The MDX syntax defines multidimensional objects and queries and manipulates multidimensional data in OLAP databases.

OLE DB and ODBC APIs Client applications use OLE DB and Open Database Connectivity (ODBC), application programming interfaces (APIs) to send commands to a database. Commands that you send through these APIs use the Transact-SQL language.

ActiveX Data Objects and ActiveX Data Objects (Multidimensional)

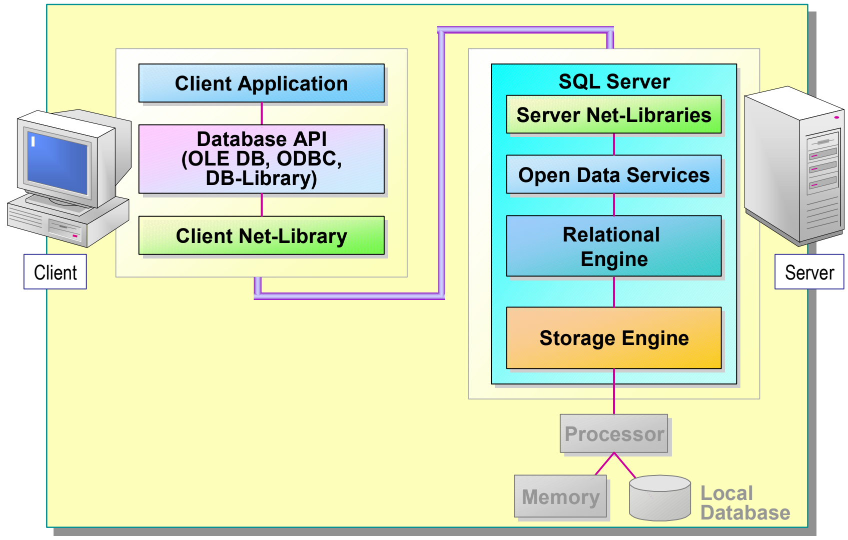
Microsoft ActiveX® Data Objects (ADO) and ActiveX Data Objects (Multidimensional) (ADO MD) wrap OLE DB for use in languages such Microsoft Visual Basic®, Visual Basic for Applications, Active Server Pages, and Microsoft Internet Explorer Visual Basic Scripting. You use ADO to access data in OLTP databases. You use ADO MD to access data in Analysis Services data cubes.

English Query This application provides an Automation API that lets users resolve natural-language questions instead of writing complex Transact-SQL or MDX statements about information in a database. For example, users are able to ask the question, "What are the total sales for Region 5?"

Client-Server Components

Topic Objective
To describe the components involved in the client-server communication process.

Lead-in
These components manage communications between clients and SQL Server.



SQL Server consists of client and server components that store and retrieve data. SQL Server uses layered communication architecture to isolate applications from the underlying network and protocols. This architecture allows you to deploy the same application in different network environments.

Client-Server Architecture

SQL Server uses client-server architecture to separate the workload into tasks that run on server computers and those that run on client computers:

- The client is responsible for business logic and presenting data to the user. The client typically runs on one or more computers, but it also can run on the server computer along with SQL Server.
- SQL Server manages databases and allocates the available server resources—such as memory, network bandwidth, and disk operations—among multiple requests.

Client-server architecture allows you to design and deploy applications to enhance a variety of environments. Client programming interfaces provide the means for applications to run on separate client computers and communicate to the server over a network.

Note In this course, the term client by itself refers to a client application.

Client Components

The client components in the communication architecture include:

Client Application A client application ends Transact-SQL statements and receives result sets. You develop an application by using a database API. The application has no knowledge of the underlying network protocols used to communicate with SQL Server.

Delivery Tip
Emphasize that the client Net-Library *must* match one of the server Net-Libraries.

Database API Database API (OLE DB, ODBC) uses a provider, driver, or DLL to pass Transact-SQL statements and receive result sets. This is an interface that an application uses to send requests to SQL Server and to process results that SQL Server returns.

Note Some Internet applications are written to HTTP rather than a database API.

Client Net-Library A client Net-Library manages network connections and routing on a client. This is a communication software component that packages the database requests and results for transmission by the appropriate network protocol.

Server Components

The server components in the communication architecture include:

Server Net-Libraries SQL Server can monitor multiple Net-Libraries concurrently. The client Net-Library must match one of the server Net-Libraries to communicate successfully. SQL Server supports network protocols such as TCP/IP, Named Pipes, NWLink, IPX/SPX, VIA ServerNet II SAN, VIA GigaNet SAN, Banyan VINES, and AppleTalk.

Open Data Services Open Data Services makes data services appear to a client as SQL Server by providing a network interface for handling network protocol processes and server routines. This is a component of SQL Server that handles network connections, passing client requests to SQL Server for processing and returning any results and replies to SQL Server clients. Open Data Services automatically listens on all server Net-Libraries that are installed on the server.

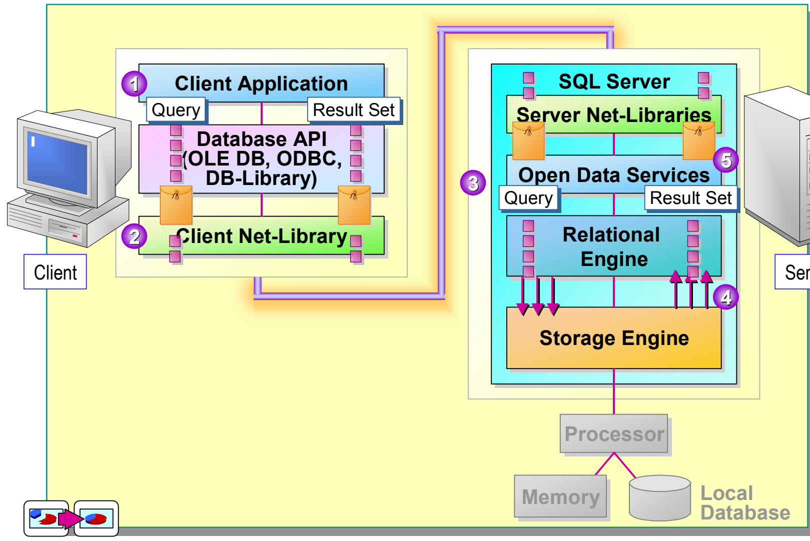
Relational Engine The relational engine parses Transact-SQL statements, optimizes and executes execution plans, processes data definition language (DDL) and other statements, and enforces security.

Storage Engine The storage engine manages database files and the use of space in the files, builds and reads data from physical pages, manages data buffers and physical input/output (I/O), controls concurrency, performs logging and recovery operations, and implements utility functions such as Database Consistency Checker (DBCC), backup, and restore.

Client-Server Communication Process

Topic Objective
To describe the client-server communication process.

Lead-in
Clients and servers typically communicate over a network.



Delivery Tip
This slide is animated. Refer to the Instructor Notes if you require help with navigating through this slide.

Point out that all communication between the client Net-Library and a SQL Server Net-Library uses the network protocol stack of the operating system.

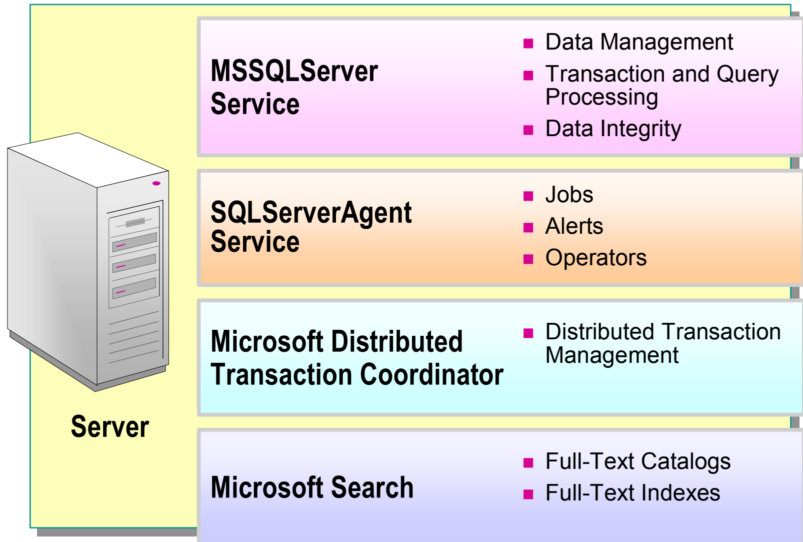
Clients and servers typically communicate over a network. The following sequence uses a query to illustrate the typical client-server communication process using a database API:

1. A client application submits a query. The client calls the database API and passes the query. The database API uses a provider, driver, or DLL to encapsulate the query in one or more Tabular Data Stream (TDS) packets and pass packets to the client Net-Library.
2. The client Net-Library packages the TDS packets into network protocol packets. The client Net-Library calls a Windows interprocess communication (IPC) API to send the network protocol packets to a server Net-Library by using the network protocol stack of the operating system. The appropriate server Net-Library extracts the TDS packets from the network protocol packets and passes the TDS packets to Open Data Services.
3. Open Data Services extracts the query from the TDS packets and passes the query to the relational engine. The relational engine then compiles the query into an optimized execution plan. It executes the execution plan. The relational engine communicates with the storage engine by using the OLE DB interface.
4. The storage engine transfers data from a database to data buffers and then passes rowsets containing data to the relational engine. The relational engine combines the rowsets into the final result set and passes the result set to Open Data Services.
5. Open Data Services packages the result set and returns it to the client application by using a server Net-Library, the network protocol stack, the client Net-Library, and the database API. The result set can also be returned in XML format.

SQL Server Services

Topic Objective
To introduce the server software for SQL Server.

Lead-in
The SQL Server services include MSSQLServer service, SQLServerAgent service, Microsoft Distributed Transaction Coordinator, and Microsoft Search.



The SQL Server services include MSSQLServer service, SQLServerAgent service, Microsoft Distributed Transaction Coordinator (MS DTC), and Microsoft Search. Although these SQL Server services usually run as services on Windows 2000, they also can run as applications.

Four SQL Server Services

SQL Server includes four services, which are installed by default with a new installation: MSSQLServer service, SQLServerAgent service, Microsoft Distributed Transaction Coordinator, and Microsoft Search.

MSSQLServer Service

MSSQLServer service is the database engine. It is the component that processes all Transact-SQL statements and manages all files that comprise the databases on the server. MSSQLServer service:

- Allocates computer resources among multiple concurrent users.
- Prevents logic problems, such as timing requests from users who want to update the same data at the same time.
- Ensures data consistency and integrity.

SQLServerAgent Service

SQLServerAgent service works in conjunction with SQL Server to create and manage alerts, local or *multiserver* jobs, and operators. Consider the following about SQLServerAgent service:

- Alerts provide information about the status of a process, such as when a job is complete or when an error occurs.
- SQLServerAgent service includes a job creation and scheduling engine that automates tasks.
- SQLServerAgent service can send e-mail messages, page an operator, or start another application when an alert occurs. For example, you can set an alert to occur when a database or transaction log is almost full or when a database backup is successful.

Microsoft Distributed Transaction Coordinator

MS DTC allows clients to include several different sources of data in one transaction. MS DTC coordinates the proper completion of distributed transactions to ensure that all updates on all servers are permanent—or, in the case of errors, that all modifications are cancelled.

Microsoft Search

Microsoft Search is a full-text engine that runs as a service in Windows 2000. Full-text support involves the ability to issue queries against character data and the creation and maintenance of the indexes that facilitate these queries.

Multiple Instances of SQL Server

Multiple instances of the SQL Server may run concurrently on the same computer.

Each instance of SQL Server has its own set of system and user databases that are not shared between instances. Each instance operates as if it were on a separate server. Applications can connect to each SQL Server database engine instance on a computer in nearly the same way that they connect to SQL Server database engines running on different computers.

When you specify only the computer name, you work with the default instance. You must specify the *computer_name\instance_name* to connect to a named instance.

◆ SQL Server Integration

Topic Objective
To outline this topic.

Lead-in
SQL Server includes client and server components.

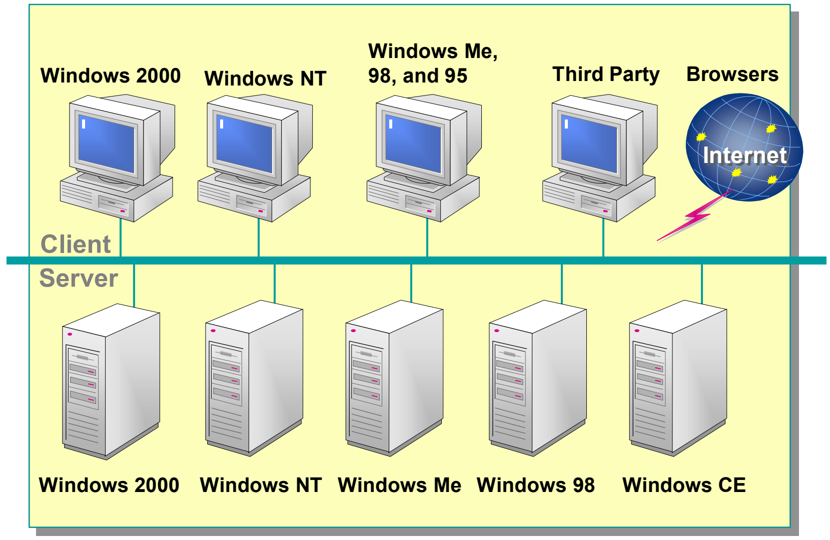
- **Integrating SQL Server with Operating Systems**
- **Integrating SQL Server with Windows 2000**
- **Integrating SQL Server with Other Microsoft Server Applications**

SQL Server includes client and server components that integrate with various Microsoft operating systems, including Windows 2000, and other Microsoft server applications. Internet browsers and third-party client applications running on various operating systems also can access SQL Server.

Integrating SQL Server with Operating Systems

Topic Objective
To describe SQL Server platforms for client and server components.

Lead-in
SQL Server includes client and server components that run on various operating systems.



SQL Server includes client and server components that run on various operating systems.

Client Components

The client components from all SQL Server 2000 editions, except SQL Server Windows CE Edition, run on all editions of Windows 2000, versions of Microsoft Windows NT®, on Microsoft Windows Millennium Edition (Me), Microsoft Windows 98, and Microsoft Windows 95.

All client components from SQL Server 2000 CE edition run exclusively on the Windows CE operating system.

Server Components

The various editions of SQL Server allow it to run on all editions of Windows 2000, versions of Windows NT, Windows Me, Windows 98, and Windows CE. Specific versions of the operating systems and editions of SQL Server limit server components. Microsoft Windows NT Server 4.0, Service Pack 5 (SP5) or later must be installed as a minimum requirement for all SQL Server 2000 editions. Only the server components, such as the database engine and the Analysis server, are limited to specific versions of the operating systems. For example, although the database engine for Microsoft SQL Server 2000 Enterprise Edition does not run on Microsoft Windows 2000 Professional, Microsoft Windows NT Workstation, Windows Me, or Windows 98, you can use the SQL Server 2000 Enterprise Edition compact disc to install the client software on any of these operating systems.

Windows NT 4.0 Terminal Server does not support SQL Server 2000.

Internet Browsers and Third-Party Applications

Internet browsers and third-party client applications running on various operating systems also can access SQL Server.

Integrating SQL Server with Windows 2000

Topic Objective
To discuss SQL Server integration with Windows 2000.

Lead-in
SQL Server is fully integrated with Windows 2000.

- Active Directory
- Security
- Multiprocessor Support
- Microsoft Event Viewer
- Windows 2000 Component Services
- Windows 2000 System Monitor
- Microsoft Internet Information Services
- Windows Clustering

SQL Server is fully integrated with Windows 2000 and takes advantage of many of its features.

Active Directory Servers and their attributes are registered automatically in the Active Directory™ directory service on server startup. Users can search and locate a particular server through Active Directory Search. For example, a user might use the directory to locate all of the servers running one or more instances of SQL Server with a particular database name installed on them.

Security SQL Server is integrated with the security system in Windows 2000. This integration allows a single user name and password to access both SQL Server and Windows 2000. SQL Server also uses encryption features in Windows 2000 for network security, including Kerberos support. SQL Server provides its own security for clients that need to access SQL Server without authentication by Windows 2000.

Multiprocessor Support SQL Server supports the symmetric multiprocessing (SMP) capabilities of Windows 2000. SQL Server automatically takes advantage of any additional processors that are added to the server computer.

Microsoft Event Viewer SQL Server writes messages to the Windows 2000 application, security, and system event logs, providing a consistent mechanism for viewing and tracking problems.

Windows 2000 Component Services Component Services is based on extensions of the Component Object Model (COM) and Microsoft Transaction Server. It provides improved threading, improved security, transaction management, object pooling, queued components, application administration, and application packaging. For example, software developers can use Component Services to visually configure routine component and application behavior, such as security and participation in transactions, and to integrate components into COM+ applications.

Windows 2000 System Monitor SQL Server sends performance metrics to the Windows 2000 System Monitor, enabling you to monitor the system performance of SQL Server.

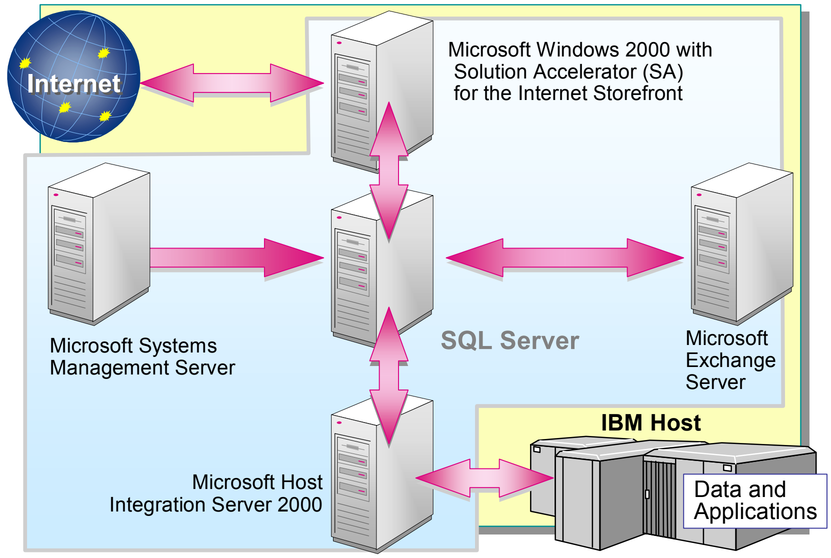
Microsoft Internet Information Services SQL Server uses Microsoft Internet Information Services (IIS) so that Internet browsers can access a SQL Server database by using the HTTP protocol.

Windows Clustering Windows Clustering, a component of Windows 2000 Advanced Server, supports the connection of two servers, or nodes, into a cluster for greater availability and better manageability of data and applications. SQL Server works in conjunction with Windows Clustering to switch automatically to the secondary node if the primary node fails.

Integrating SQL Server with Other Microsoft Server Applications

Topic Objective
To show how SQL Server works with other Microsoft server applications.

Lead-in
SQL Server integrates well with other Microsoft server applications.



SQL Server integrates well with other Microsoft server applications. Microsoft provides a group of server applications that work together to help you build business solutions. The following table describes some commonly used server applications that work with or use SQL Server.

Server application	Description
Microsoft Windows 2000 Server with Solution Accelerator (SA) for the Internet Storefront	Provides secure, fast, manageable Internet connectivity. Includes an extensible, multilevel enterprise firewall and scalable high-performance Web cache.
Microsoft Exchange Server	Allows SQL Server to send e-mail messages by using Exchange Server or other Messaging Application Programming Interface (MAPI)-compliant providers. SQL Server can send messages when an error occurs or a scheduled task (such as a database backup) succeeds or fails. It also can respond to queries embedded in messages.
Microsoft Host Integration Server 2000	Links IBM environments running the Systems Network Architecture (SNA) protocol with PC-based networks. You can integrate SQL Server with IBM mainframe or AS/400 applications and data by using Microsoft Host Integration Server 2000.
Microsoft Systems Management Server	Manages computer software, hardware, and inventory and uses SQL Server to store its databases.

◆ SQL Server Databases

Topic Objective
To introduce the topics on SQL Server databases.

Lead-in
An understanding of SQL Server database structure will help you develop and implement your database effectively.

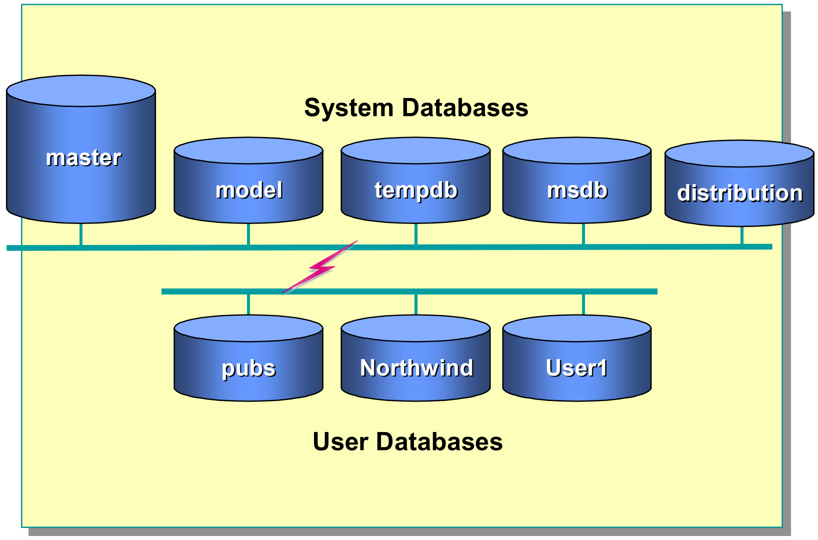
- **Types of Databases**
- **Database Objects**
- **Referring to SQL Server Objects**
- **System Tables**
- **Metadata Retrieval**

An understanding of SQL Server database structure will help you develop and implement your database effectively.

Types of Databases

Topic Objective
To illustrate the two types of SQL Server databases.

Lead-in
Each SQL Server has two types of databases: system databases and user databases.



Each SQL Server has two types of databases: *system databases* and *user databases*. System databases store information about SQL Server as a whole. SQL Server uses system databases to operate and manage the system. User databases are databases that users create.

Delivery Tip
Briefly describe the function of each type of database. Point out that SQL Server supports multiple instances.

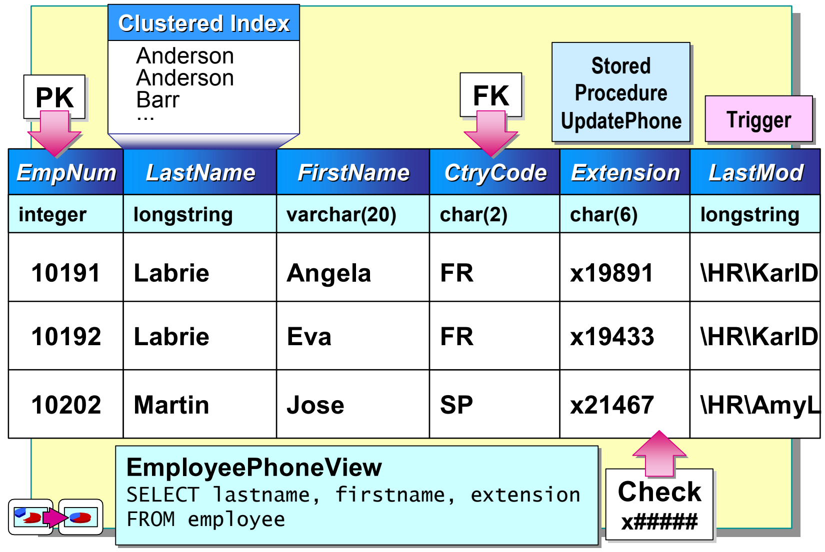
When SQL Server is installed, SQL Server Setup creates system databases and sample user databases. The Distribution Database is installed when you configure SQL Server for replication activities. The following table describes each database.

Database	Description
master	Controls the user databases and operation of SQL Server as a whole by keeping track of information such as user accounts, configurable environment variables, and system error messages
model	Provides a template or prototype for new user databases
tempdb	Provides a storage area for temporary tables and other temporary working storage needs
msdb	Provides a storage area for scheduling information and job history
distribution	Stores history and transaction data used in replication
pubs	Provides a sample database as a learning tool
Northwind	Provides a sample database as a learning tool
User1	Identifies a user-defined database

Database Objects

Topic Objective
To describe the database objects related to SQL Server.

Lead-in
A database is a collection of data, tables, and other objects.



Delivery Tip
This slide is animated. Refer to the Instructor Notes if you require help with navigating through this slide.

Note that the information in the illustration is fictitious and is provided as a teaching example.

Explain that the list of database objects in the table is generic and does not represent an object hierarchy.

Briefly describe each database object.

A database is a collection of data, tables, and other objects. Database objects help you structure data and define data integrity mechanisms. The following table describes SQL Server database objects.

Database object	Description
Table	Defines a collection of rows that have associated columns.
Data type	Defines the data values allowed for a column or variable. SQL Server provides system-supplied data types. Users create user-defined data types.
Constraint	Defines rules regarding the values allowed in columns and is the standard mechanism for enforcing data integrity.
Default	Defines a value that is stored in a column if no other value is supplied.
Rule	Contains information that defines valid values that are stored in a column or data type.
Index	Is a storage structure that provides fast access for data retrieval and can enforce data integrity. In a clustered index, the logical or indexed order of the key values is the same as the physical, stored order of the corresponding rows that exist in the table. In a nonclustered index, the logical order of the index does not match the physical, stored order of the rows in the table.

(*continued*)

Database object	Description
View	Provides a way to look at data from one or more tables or views in a database.
User-defined function	Can return either a scalar value or a table. Functions are used to encapsulate frequently performed logic. Any code that must perform the logic incorporated in a function can call the function rather than having to repeat all of the function logic.
Stored procedure	Is a named collection of precompiled Transact-SQL statements that execute together.
Trigger	Is a special form of a stored procedure that is executed automatically when a user modifies data in a table or a view.

Referring to SQL Server Objects

Topic Objective
To introduce the SQL Server object-naming standard.

Lead-in
You can refer to SQL Server objects in several ways…

- **Fully Qualified Names**

  server.database.owner.object

- **Partially Specified Names**
  - Server defaults to the current instance on the local server
  - Database defaults to current database
  - Owner defaults to the user name in the database

```
CREATE TABLE Northwind.dbo.OrderHistory
.
.
.
```

You can refer to SQL Server objects in several ways. You can specify the full name of the object (its fully qualified name), or specify only part of the name of the object name and allow SQL Server to determine the rest of the name from the context in which you are working.

Fully Qualified Names

The complete name of a SQL Server object includes four identifiers—the server name, database name, owner name, and object name in the following format:

server.database.owner.object

An object name that specifies all four parts is known as a fully qualified name. Each object that you create in SQL Server must have a unique, fully qualified name. For example, you can have two tables named **Orders** in the same database as long as they belong to different owners. Also, column names must be unique within a table or view.

Partially Specified Names

When referencing an object, you do not always have to specify the server, database, and owner. Intermediate identifiers can be omitted as long as their positions are indicated by periods.

The following list contains valid formats for object names:

server.database.owner.object
database.owner.object
database..object
owner.object
object

When you create an object and do not specify the different parts of the name, SQL Server uses the following defaults:

- Server defaults to the current instance on the local server.
- Database defaults to the current database.
- Owner defaults to the user name in the specified database associated with the login ID of the current connection.

A user that is a member of a role can explicitly specify the role as the object owner. A user that is a member of the **db_owner** or **db_ddladmin** role in a database should specify the **dbo** user account as the owner of an object. This practice is recommended.

Example

The following example creates an **OrderHistory** table in the **Northwind** database.

```
CREATE TABLE Northwind.dbo.OrderHistory
(OrderID int,
ProductID int,
UnitPrice money,
Quantity int,
Discount decimal)
```

Most object references use three-part names and default to the local server. Four-part names are generally used for distributed queries or remote stored procedure calls.

SQL Server supports a three-part naming convention when referring to the current server. The SQL-92 standard also supports a three-part naming convention. The terms used in both naming conventions are different. The following table describes the relationships between SQL Server names and SQL-92-standard names.

SQL Server name	SQL-92 name
Database	catalog
Owner	schema
Object	object

System Tables

Topic Objective
To describe system tables.

Lead-in
System tables store information, called metadata, about the system and objects in databases.

- **System Tables Store Information (Metadata) About the System and Database Objects**
- **Database Catalog Stores Metadata About a Specific Database**
- **System Catalog Stores Metadata About the Entire System and All Other Databases**

Delivery Tip
Point students to SQL Server Books Online for detailed discussion of metadata.

SQL Server stores information, called metadata, about the system and objects in databases for an instance of SQL Server. *Metadata* is information about data.

Metadata includes information about the properties of data, such as the type of data in a column (numeric, text, and so on), or the length of a column. It can also be information about the structure of data or information that specifies the design of objects.

System Tables The information about data in system tables includes configuration information and definitions of all of the databases and database objects in the instance of SQL Server. Users should not directly modify any system table.

Database Catalog Each database (including **master**) contains a collection of system tables that store metadata about that specific database. This collection of system tables is the database catalog. It contains the definition of all of the objects in the database, as well as permissions.

System Catalog The system catalog, found only in the **master** database, is a collection of system tables that stores metadata about the entire system and all other databases.

Most system tables begin with the **sys** prefix. The following table identifies several frequently used system tables and views.

System table	Database	Function
syslogins	**master**	Contains one row for each login account that can connect to SQL Server
sysmessages	**master**	Contains one row for each system error or warning that SQL Server can return
sysdatabases	**master**	Contains one row for each database on SQL Server
sysusers	All	Contains one row for each Windows 2000 user, Windows 2000 group, SQL Server user, or SQL Server role in a database
sysobjects	All	Contains one row for each object in a database

Metadata Retrieval

Topic Objective
To discuss the different ways to retrieve information from system tables.

Lead-in
When you write applications that retrieve information from system tables, you should use…

- **System Stored Procedures**

```
EXEC sp_help Employees
```

- **System and Metadata Functions**

```
SELECT USER_NAME(10)
```

- **Information Schema Views**

```
SELECT * FROM INFORMATION_SCHEMA.TABLES
```

When you write applications that retrieve metadata from system tables, you should use system stored procedures, system functions, or system-supplied information schema views.

Delivery Tip
Emphasize that writing scripts that directly query system tables is not recommended.

If system tables are changed in future product versions, scripts that rely on particular system table structures may fail or may not provide accurate information.

Use information schema views instead of system tables.

You can query a system table in the same way that you do any other database table to retrieve information about the system. However, you should not write scripts that directly query system tables, because if the system tables change in future product versions, your scripts may fail or may not provide accurate information.

Warning You should not alter system tables directly. Changing a system table may make it impossible for SQL Server to recover properly in the event of a system failure.

System Stored Procedures

To make it easier for you to gather information about the state of the server and database objects, SQL Server provides a collection of prewritten queries called system stored procedures.

The names of most system stored procedures begin with the **sp_** prefix. The following table describes three commonly used system stored procedures.

System stored procedure	Description
sp_help [*object_name*]	Provides information on the specified database object
sp_helpdb [*database_name*]	Provides information on the specified database
sp_helpindex [*table_name*]	Provides information on the index for the specified table

Example

The following example executes a system stored procedure to get information on the **Employees** table.

```
EXEC sp_help Employees
```

Delivery Tip
Point out that system and metadata functions return specific, single values.

System and Metadata Functions

System and metadata functions provide a method for querying system tables from within Transact-SQL statements. The following table describes commonly used system functions and the corresponding information returned.

System function	Parameter passed	Results
DB_ID	Name	Returns the database ID
USER_NAME	ID	Returns the user's name
COL_LENGTH	Column	Returns the column width
STATS_DATE	Index	Returns the date when statistics for the specified index were last updated
DATALENGTH	Data type	Returns the actual length of an expression of any data type

Example 1

The following example uses a system function in a query to retrieve the user name for a user ID of 10.

```
SELECT USER_NAME(10)
```

Information Schema Views

Information schema views provide an internal, system table-independent view of the SQL Server metadata. These views conform to the ANSI SQL standard definition for the information schema.

Each information schema view contains metadata for all data objects stored in that particular database. The following table describes commonly used information schema views.

Information schema view	Description
INFORMATION_SCHEMA.TABLES	List of tables in the database
INFORMATION_SCHEMA.COLUMNS	Information on columns defined in the database
INFORMATION_SCHEMA.TABLES_PRIVILEGES	Security information for tables in the database

Example 2

The following example queries an information schema view to retrieve a list of tables in a database.

```
SELECT * FROM INFORMATION_SCHEMA.TABLES
```

◆ SQL Server Security

Topic Objective
To describe the security for SQL Server.

Lead-in
SQL Server validates users at two levels of security—login authentication, and permissions validation on database user accounts and roles.

- **Login Authentication**
- **Database User Accounts and Roles**
- **Types of Roles**
- **Permission Validation**

SQL Server validates users at two levels of security—login authentication, and permissions validation on database user accounts and roles.

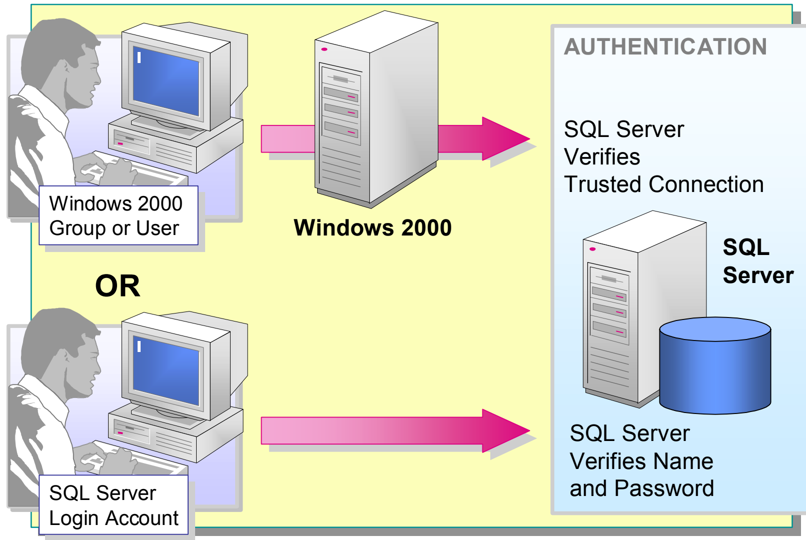
Authentication identifies the user who is using a login account and verifies the user's the ability to connect with SQL Server. If authentication is successful, the user connects to SQL Server.

The user then must have permission to access databases on the server. The database administrator assigns database-specific permissions to user accounts and roles in order to access databases on the server. Permissions control the activities that the user is allowed to perform in the SQL Server database.

Login Authentication

Topic Objective
To introduce how SQL Server authenticates login accounts.

Lead-in
A user must have a login account to connect to SQL Server.



A user must have a login account to connect to SQL Server. SQL Server recognizes two login authentication mechanisms—Windows Authentication and SQL Server Authentication—each of which has a different type of login account.

Windows Authentication

When using Windows Authentication, a Windows 2000 account or group controls user access to SQL Server. A user does not provide a SQL Server login account when connecting. A SQL Server system administrator must define either the Windows 2000 account or the Windows 2000 group as a valid SQL Server login account.

SQL Server Authentication

When using SQL Server Authentication, a SQL Server system administrator defines a SQL Server login account and password. Users must supply both SQL Server logins and passwords when they connect to SQL Server.

Delivery Tip
Because students may find SQL Server Authentication modes and mechanisms confusing, point out that the server runs in an authentication mode and that a client connecting to a server running in Mixed Mode must choose an authentication mechanism.

Authentication Mode

When SQL Server is running on Windows 2000, a system administrator can specify that it run in one of two authentication modes:

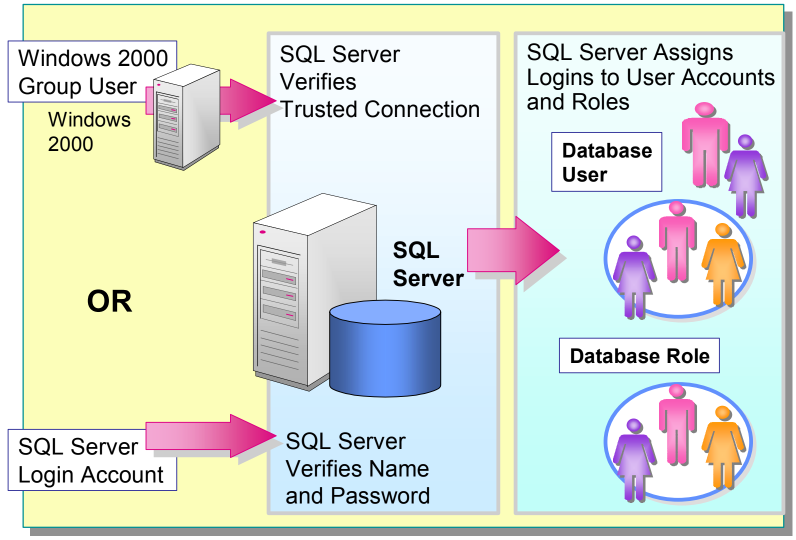
Windows Authentication Mode Only Windows 2000 authentication is allowed. Users cannot specify a SQL Server login account.

Mixed Mode Users can connect to SQL Server with Windows Authentication or SQL Server Authentication.

Database User Accounts and Roles

Topic Objective
To introduce how SQL Server matches SQL Server login authentication to user accounts and roles.

Lead-in
After users have been authenticated and have been allowed to log in to SQL Server, they must have accounts in a database.



After users have been authenticated by Windows 2000 or SQL Server and have been allowed to log in to SQL Server, they must have accounts in a database. User accounts and roles identify a user within a database and control ownership of objects and permissions to execute statements.

Database User Accounts

The user accounts that apply security permissions are Windows 2000 users or groups or SQL Server login accounts. User accounts are specific to a database.

Roles

Roles enable you to assemble users into a single unit to which you can apply permissions. SQL Server provides predefined server and database roles for common administrative functions so that you can easily grant a selection of administrative permissions to a particular user. You also can create your own user-defined database roles. In SQL Server, users can belong to multiple roles.

Types of Roles

Topic Objective
To describe three types of roles.

Lead-in
SQL Server enables three types of roles.

- **Fixed Server Roles**
  - Group administrative privileges at the server level
- **Fixed Database Roles**
  - Group administrative privileges at the database level
- **User-defined Database Roles**
  - Represent work defined by a group of employees within an organization

SQL Server enables three types of roles to help manage permissions: fixed server roles, fixed database roles, and user-defined database roles.

Fixed Server Role

Fixed server roles provide groupings of administrative privileges at the server level. They are managed independently of user databases at the server level. The following table describes the fixed server roles in SQL Server 2000.

Role	Permission
Database creators **(dbcreator)**	Create and alter databases
Disk administrators **(diskadmin)**	Manage disk files
Process administrators **(processadmin)**	Manage SQL Server processes
Security administrators **(securityadmin)**	Manage and audit server logins
Server administrators **(serveradmin)**	Configure server-wide settings
Setup administrators **(setupadmin)**	Install replication
System administrators **(sysadmin)**	Perform any activity
Bulk administrators **(bulkadmin)**	Execute BULK INSERT statement

Fixed Database Roles

Fixed database roles provide groupings of administrative privileges at the database level. The following table describes the fixed database roles in SQL Server 2000.

Role	Permission
public	Maintain all default permissions for users in a database
db_owner	Perform any database role activity
db_accessadmin	Add or remove database users, groups, and roles
db_ddladmin	Add, modify, or drop database objects
db_securityadmin	Assign statement and object permissions
db_backupoperator	Back up databases
db_datareader	Read data from any table
db_datawriter	Add, change, or delete data from all tables
db_denydatareader	Cannot read data from any table
db_denydatawriter	Cannot change data in any table

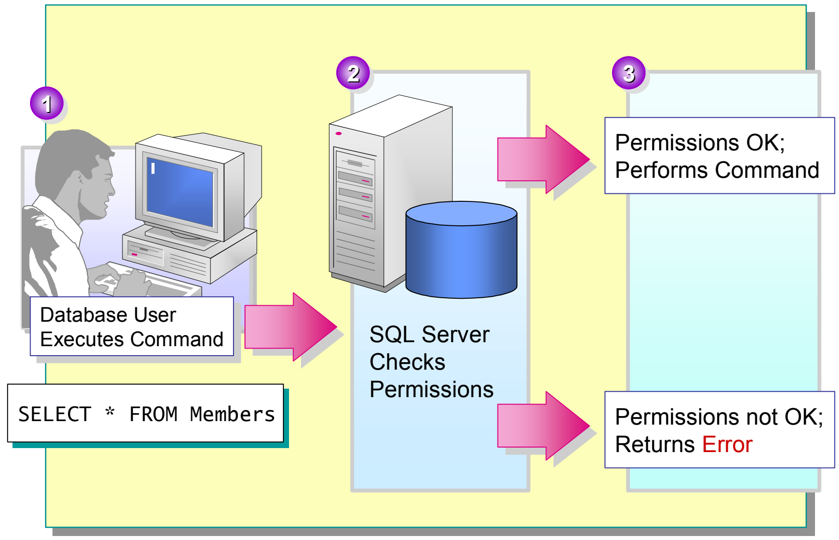
User-defined Database Roles

You also can create your own database roles to represent work performed by a group of employees in your organization. You do not have to grant and revoke permissions from each person. If the function of a role changes, you easily can change the permissions for the role and have the changes apply automatically to all members of the role.

Permission Validation

Topic Objective
To describe how SQL Server validates permissions.

Lead-in
Within each database, you assign permissions to user accounts and roles to perform (or restrict) certain actions.



Within each database, you assign permissions to user accounts and roles to perform (or restrict) certain actions. SQL Server accepts commands after a user has successfully accessed a database.

SQL Server takes the following steps when validating permissions:

1. When the user performs an action, such as executing a Transact-SQL statement or choosing a menu option, the client sends Transact-SQL statements to SQL Server.
2. When SQL Server receives a Transact-SQL statement, it checks that the user has permission to execute the statement.
3. SQL Server then performs one of two actions:
   - If the user does not have the proper permissions, SQL Server returns an error.
   - If the user has the proper permissions, SQL Server performs the action.

◆ Working with SQL Server

Topic Objective
To review the following topics: SQL Server application design and SQL Server database implementation and administration.

Lead-in
Working with SQL Server involves administering and implementing a SQL Server database and developing applications.

- Administering a SQL Server Database
- Implementing a SQL Server Database
- Selecting an Application Architecture for SQL Server
- Designing Applications Using Database APIs

Working with SQL Server involves administering and implementing a SQL Server database and developing applications.

Delivery Tip
Discuss when to use a script versus a graphical tool to work with SQL Server.

When you work with SQL Server, you can use graphical tools that SQL Server provides, or you can use scripts. For example, when you need to create a database, you can use scripts or a graphical tool. Because generally you only create a particular database once, you would most likely use a graphical tool. However, when backing up the database, you would most likely use a script, because backing up is a task that must be done repeatedly.

Administering a SQL Server Database

Topic Objective
To introduce SQL Server administration activities.

Lead-in
SQL Server provides graphical and command prompt tools and utilities to administer SQL Server.

- **Common Administrative Tasks**
- **SQL Server Enterprise Manager**
- **SQL Server Administration Tools and Wizards**
- **SQL Server Command Prompt Management Tools**
- **SQL Server Help and SQL Server Books Online**

SQL Server provides graphical and command prompt tools and utilities to administer SQL Server. It also includes different types of Help to assist you.

Common Administrative Tasks

Administering a SQL Server database involves:

- Installing, configuring, and securing SQL Server.
- Building databases.

  Tasks include allocating disk space to the database and log, transferring data into and out of the database, defining and implementing database security, creating automated jobs for repetitive tasks, and setting up replication to publish data to multiple sites.
- Managing ongoing activities, such as importing and exporting data, backing up and restoring the database and log, and monitoring and tuning the database.

Delivery Tip
Briefly demonstrate SQL Server Enterprise Manager, SQL Query Analyzer, and the **osql** utility.

SQL Server includes tools and wizards for administering and managing the server, designing and creating databases, and querying data. It also provides online Help.

SQL Server Enterprise Manager

SQL Server provides an administrative client, SQL Server Enterprise Manager, which is a Microsoft Management Console (MMC) snap-in. MMC is a shared user interface for the management of Microsoft server applications.

SQL Server Administration Tools and Wizards

SQL Server provides a number of administrative tools and wizards that assist with particular aspects of its administration. The following table describes SQL Server tools and wizards.

Graphical tool	Purpose
Client Network Utility	Utility for managing the client configuration for network libraries
SQL Server Network Utility	Utility for managing the server configuration for network libraries
SQL Profiler	Utility for capturing a continuous record of server activity and providing auditing capability
SQL Query Analyzer	Graphical query tool for analyzing the plan of a query, viewing statistics information, and managing multiple queries in different windows simultaneously
SQL Server Service Manager	Graphical utility for starting, stopping, and pausing SQL Server services
SQL Server Setup	Application for installing and configuring SQL Server
SQL Server wizards	Collection of tools that guide users through complex tasks

SQL Server Command Prompt Management Tools

SQL Server command prompt management tools allow you to enter Transact-SQL statements and execute script files. The following table describes the most frequently used command prompt utilities that are provided with SQL Server. Each file is an executable application.

Utility	Description
osql	Utility that uses Open Database Connectivity (ODBC) to communicate with SQL Server—primarily used to execute batch files containing one or more SQL statements
bcp	Batch utility used to import and export data to and from SQL Server—copies data to or from a data file in a user-specified format

Delivery Tip
Briefly describe each type of Help that SQL Server provides.

SQL Server Help and SQL Server Books Online

SQL Server offers different types of Help to assist you. The following table describes each type of Help that SQL Server provides.

Type of Help	Description
Tool Help	SQL Server tools generally provide context-sensitive help on the application interface. Click the **Help** button or a command on the **Help** menu.
Transact-SQL Help	When using SQL Query Analyzer, select a statement name and then press SHIFT+F1.
SQL Server documentation set	SQL Server Books Online provides online access to SQL Server documentation.

Implementing a SQL Server Database

Topic Objective
To introduce SQL Server implementation activities.

Lead-in
Implementing a SQL Server database means planning, creating, and maintaining a number of interrelated components.

- **Designing the Database**
- **Creating the Database and Database Objects**
- **Testing and Tuning the Application and Database**
- **Planning Deployment**

Implementing a SQL Server database means planning, creating, and maintaining a number of interrelated components.

The nature and complexity of a database application, as well as the process of planning it, can vary greatly. For example, a database can be relatively simple, designed for use by a single person, or it can be large and complex, designed to handle all the banking transactions for hundreds of thousands of clients.

Delivery Tip
Briefly review each step.

Regardless of the size and complexity of the database, implementing it usually involves:

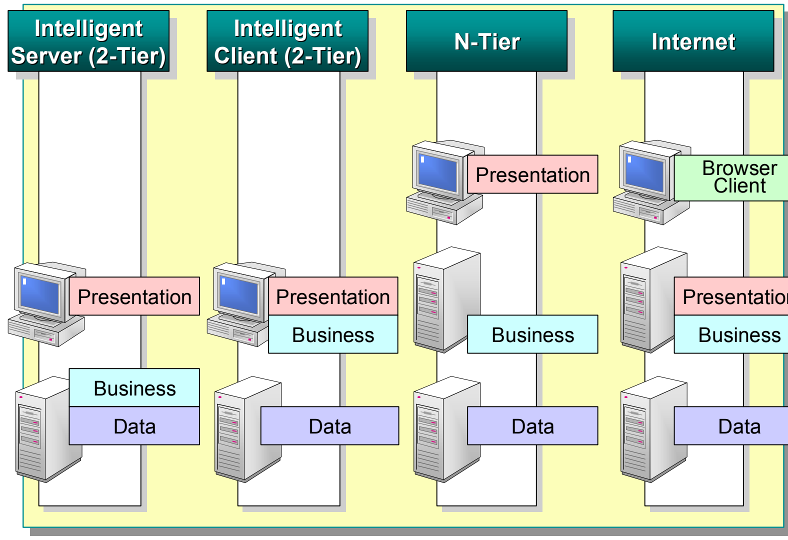
- Designing the database so that your application uses hardware optimally and allows for future growth; identifying and modeling database objects and application logic; and specifying the types of information for each object and type of relationship.
- Creating the database and database objects, including tables, data integrity mechanisms, data entry and retrieval objects (often stored procedures), appropriate indexes, and security.
- Testing and tuning the application and database.

  When you design a database, you want to ensure that the database performs important functions correctly and quickly. In conjunction with correct database design, the correct use of indexes, RAID, and filegroups are essential to achieving good performance.
- Planning deployment, which includes analyzing the workload and recommending an optimal index configuration for your SQL Server database.

Selecting an Application Architecture for SQL Server

Topic Objective
To describe the physical architectures and logical software application designs available when planning a database.

Lead-in
Planning a database design requires knowledge of the business functions that you want to model and the database concepts and features that you can use to represent those business functions.



Planning a database design requires knowledge of the business functions that you want to model and the database concepts and features that you use to represent those business functions.

Before you design an application for SQL Server, it is important to take time designing a database to model the business accurately. A well-designed database requires fewer changes and generally performs more efficiently. The architecture that you select affects how you develop, deploy, and manage your software application.

Delivery Tip
Point out that only the most common architectural designs and deployment options are illustrated.

Emphasize that in any architectural design, the software application can reside physically on one server.

Software Architecture

You can use one of several application architectures to implement client/server applications. However, selecting a layered application approach affords flexibility and a choice of management options. You can divide software applications into three logical layers, which can physically reside on one or more servers.

Logical layer	Description
Presentation	Includes the logic for presenting the data and application to users. This layer is almost always implemented on a client computer.
Business	Includes the application logic and business rules. (SQL Server can be involved with this layer.)
Data	Includes database definition, data integrity logic, stored procedures, and other operations that are closely associated with the data. (SQL Server is involved primarily with this layer.)

Delivery Tip
Briefly describe each application deployment option and point out the examples.

Architectural Design

Typical application deployment options include:

Intelligent Server (2-Tier) Most processing occurs on the server, with the client handling presentation services. In many instances, most of the business services logic is implemented in the database. This design is useful when clients do not have sufficient resources to process the business logic. However, the server can become a bottleneck because database and business services compete for the same hardware resources.

Corporate applications designed from a database-centric point of view are an example of this design.

Intelligent Client (2-Tier) Most processing occurs on the client, with the server handling data services. This design is widely used. However, network traffic can be heavy and transactions longer, which can affect performance.

Applications developed for small organizations with products such as Microsoft Access are an example of this design.

N-Tier Processing is divided among a database server, an application server, and clients. This approach separates logic from data services, and you easily can add more application servers or database servers as needed. However, the potential for complexity increases, and this approach may be slower for small applications.

Multitiered enterprise applications and applications developed with transaction processing monitors are examples of this design.

Internet Processing is divided into three layers, with the business and presentation services residing on the Web server and the clients using Internet browsers. SQL Server uses XML support for presentation of data to browsers. SQL Server can support any client that has a browser, and software does not need to be maintained on the client.

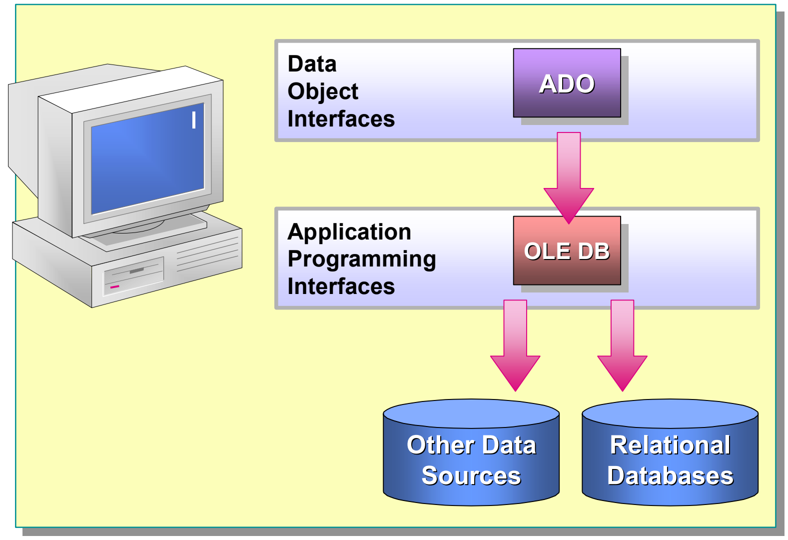
An example of this design is a Web site that uses several Web servers to manage connections to clients and a single SQL Server database that services requests for data.

You can access SQL Server over HTTP by using a URL. This allows you to directly access database objects and execute template files. This is not recommended for environments that must be highly secure and in which performance is critical.

Designing Applications Using Database APIs

Topic Objective
To describe how to design an application for SQL Server by using database APIs.

Lead-in
You can develop a database application to access SQL Server through an API.



You can develop a database application that accesses SQL Server through an API. A database API contains two parts:

- Transact-SQL language statements passed to the database.
- A set of functions or object-oriented interfaces and methods used to send the Transact-SQL statements to the database and process the results returned by the database.

Examples of relational database applications include data entry applications for airline ticketing and banking transaction systems.

Delivery Tip
Refer students to SQL Server Books Online for a detailed discussion of programming interfaces.

OLE DB

OLE DB is a Component Object Model (COM)-based API. This API is a library of COM interfaces that enables universal access to diverse data sources.

SQL Server includes a native OLE DB provider. The provider supports applications written by using OLE DB, or other APIs that use OLE DB, such as ADO. Through the native provider, SQL Server also supports objects or components using OLE DB, such as ActiveX, ADO, or Microsoft .NET Enterprise Servers.

ADO

This database API defines how to write an application to connect to a database by using OLE DB and how to pass Transact-SQL commands to a database.

ADO is an application-level interface that uses OLE DB. Because ADO uses OLE DB as its foundation, it benefits from the data access infrastructure that OLE DB provides, yet it shields the application developer from the necessity of programming COM interfaces. Developers can use ADO for general-purpose access programs in business applications (Accounting, Human Resources, and Customer Management), and can use OLE DB for tool, utility, or system-level development (development tools and database utilities).

Lab A: SQL Server Overview

Topic Objective
To introduce the lab.

Lead-in
In this lab, you will use SQL Server Books Online to find information about the **Northwind** database, and then you will create a **Northwind** database diagram.

Explain the lab objectives.

Objectives

After completing this lab, you will be able to:

- View the contents, use the index, and search for information in Microsoft® SQL Server™ Books Online, as well as save the location of information on the **Favorites** tab.
- Create a **Northwind** database diagram.

Prerequisites

None.

Lab Setup

None.

For More Information

If you require help with executing files, search SQL Query Analyzer Help for "Execute a query".

Other resources that you can use include:

- The **Northwind** database schema.
- SQL Server Books Online.

Scenario

The organization of the classroom is meant to simulate a worldwide trading firm named Northwind Traders. Its fictitious domain name is nwtraders.msft. The primary DNS server for nwtraders.msft is the instructor computer, which has an Internet Protocol (IP) address of 192.168.*x*.200 (where *x* is the assigned classroom number). The name of the instructor computer is London.

The following table provides the user name, computer name, and the IP address for each student computer in the fictitious nwtraders.msft domain. Find the user name for your computer and make a note of it.

User name	Computer name	IP address
SQLAdmin1	Vancouver	192.168.*x*.1
SQLAdmin2	Denver	192.168.*x*.2
SQLAdmin3	Perth	192.168.*x*.3
SQLAdmin4	Brisbane	192.168.*x*.4
SQLAdmin5	Lisbon	192.168.*x*.5
SQLAdmin6	Bonn	192.168.*x*.6
SQLAdmin7	Lima	192.168.*x*.7
SQLAdmin8	Santiago	192.168.*x*.8
SQLAdmin9	Bangalore	192.168.*x*.9
SQLAdmin10	Singapore	192.168.*x*.10
SQLAdmin11	Casablanca	192.168.*x*.11
SQLAdmin12	Tunis	192.168.*x*.12
SQLAdmin13	Acapulco	192.168.*x*.13
SQLAdmin14	Miami	192.168.*x*.14
SQLAdmin15	Auckland	192.168.*x*.15
SQLAdmin16	Suva	192.168.*x*.16
SQLAdmin17	Stockholm	192.168.*x*.17
SQLAdmin18	Moscow	192.168.*x*.18
SQLAdmin19	Caracas	192.168.*x*.19
SQLAdmin20	Montevideo	192.168.*x*.20
SQLAdmin21	Manila	192.168.*x*.21
SQLAdmin22	Tokyo	192.168.*x*.22
SQLAdmin23	Khartoum	192.168.*x*.23
SQLAdmin24	Nairobi	192.168.*x*.24

Estimated time to complete this lab: 30 minutes

Exercise 1
Using SQL Server Books Online

In this exercise, you will use SQL Server Books Online to retrieve information on SQL Server.

► To view the contents of Getting Started in SQL Server Books Online

In this procedure, you will view the contents of SQL Server Books Online and familiarize yourself with conventions used in the documentation.

1. Log on to the **NWTraders** classroom domain by using the information in the following table.

Option	Value
User name	**SQLAdmin*x*** (where *x* corresponds to your computer name as designated in the nwtraders.msft classroom domain)
Password	**password**

2. On the taskbar, click the **Start** button, point to **Programs**, point to **Microsoft SQL Server**, and then click **Books Online**.

   Note that you can also access SQL Server Books Online directly from the SQL Server 2000 compact disc. Insert the SQL Server 2000 compact disc into the CD-ROM drive, and when the **Microsoft SQL Server** dialog box appears, click **Browse Books Online**.

3. In the console tree, review the organization of SQL Server Books Online.
4. On the **Contents** tab, in the **Active Subset** list, click **Entire Collection**, and then review the contents of **Getting Started**.
5. In the console tree, expand **Getting Started with SQL Server Books Online**, and then click **Documentation Conventions**. Review the information in the details pane.
6. In the console tree, expand **Using SQL Server Books Online**, and then click **Finding a Topic**. Review the information in the details pane.
7. In the console tree, expand **Finding a Topic**, and then click **Using the Search tab**. Review the information in the details pane.

► To use the SQL Server Books Online index to obtain information on the Northwind sample database

In this procedure, you will use the SQL Server Books Online index to view information on the **Northwind** sample database quickly.

1. Click the **Index** tab, and then type **Northwind**
2. Double-click **Northwind sample database**.
3. In the **Topics Found** dialog box, double-click **Northwind sample database**. Review the information in the details pane.
4. Click the **Favorites** tab, and then click **Add**.

   Click the **Contents** tab, and then in the console tree, expand **Northwind sample database** and notice the available topics.

► To search SQL Server Books Online for a word or phrase

In this procedure, you will use SQL Server Books Online to search for information about the architecture of SQL Server.

1. Click the **Search** tab, type **sql NEAR architecture** and then click **List Topics**.

   Notice the number of topics that are found.

2. On the **Search** tab, clear the **Match similar words** check box, select the **Search titles only** check box, and then click **List Topics**.

   Notice that only two topics are found.

3. Double-click **Fundamentals of SQL Server Architecture**.
4. Click the details pane, and then press CTRL+F.
5. In the **Find** box, type **oltp** and then click **Find Next**.

   Notice that the search finds the first instance of oltp.

6. Close SQL Server Books Online.

Exercise 2
Creating a Database Diagram

In this exercise, you will use the Create Database Diagram Wizard to generate a diagram for the **Northwind** database automatically. You will then use Query Designer to build a query.

► **To create a database diagram**

In this procedure, you will create a database diagram of the **Northwind** database by using the Create Database Diagram Wizard.

1. On the taskbar, click the **Start** button, point to **Programs**, point to **Microsoft SQL Server**, and then click **Enterprise Manager**.
2. In the console tree, expand **Microsoft SQL Servers**, expand **SQL Server Group**, and then expand your server.
3. Expand **Databases**, and then expand **Northwind**.
4. Right-click **Diagrams**, and then click **New Database Diagram**.

   The **Welcome to the Create Database Diagram Wizard** dialog box appears.
5. Using the Create Database Diagram wizard, add the following tables to the diagram:
   - **Categories**
   - **CustomerCustomerDemo**
   - **CustomerDemographics**
   - **Customers**
   - **Employees**
   - **EmployeeTerritories**
   - **Order Details**
   - **Orders**
   - **Products**
   - **Region**
   - **Shippers**
   - **Suppliers**
   - **Territories**
6. On the toolbar, click **Zoom**, and then click **100%**.
7. On the toolbar, click **Save**.
8. In the **Save As** dialog box, type **Northwind** and then click **OK**.

► To build a query by using Query Designer

In this procedure, you will use Query Designer to build a query joining the **Products** and **Categories** tables.

1. In the diagram, right-click **Products**, point to **Task**, and then click **Open Table**.

   This opens a separate window and displays all rows in the **Products** table.

2. On the toolbar, click **Show/Hide Diagram Pane**.

   This will display the table diagram of the **Products** table.

3. Right-click an open area of the diagram pane, and then click **Add Table**.
4. Double-click **Categories**, and then click **Close**.

   This will add the **Categories** table to the diagram pane and display the relationship between the **Categories** and **Products** tables.

5. On the toolbar, click **Show/Hide SQL Pane**.

   This will display the Transact-SQL script that has been created for you.

6. In the **Products** table, select the **ProductName** and **UnitPrice** check boxes, and in the **Categories** table, select the **CategoryName** check box.
7. In the **Products** table, click **UnitPrice**, and then on the toolbar, click **Sort descending**.

   Notice the modifications that occur to the Transact-SQL script.

8. In the SQL pane, delete the asterisk and comma in the SELECT statement.
9. On the toolbar, click **Run**.

   What is the highest-priced product?

   Côte de Blaye, 263.5

   ____

   ____

Note Query Designer is running as a Microsoft Management Console (MMC) snap-in. Note that you now have three active MMC windows—SQL Server Enterprise Manager, Database Designer, and Query Designer.

Review

Topic Objective
To reinforce module objectives by reviewing key points.

Lead-in
The review questions cover some key concepts taught in the module.

- **What Is SQL Server?**
- **SQL Server Integration**
- **SQL Server Databases**
- **SQL Server Security**
- **Working with SQL Server**

Use the questions provided here to review module topics.

Before continuing, ask students whether they have any other questions.

1. You have created a new login account within SQL Server for a Windows 2000 group. You want the members of this group to be able to use SQL Server. What additional security tasks must you perform?

   You must add the Windows 2000 group to the database to which you want the users to access, and then grant permissions on the database objects to which you want the users to access.

2. You want to view metadata about objects in a SQL Server database. What methods would you use?

   You could query the information schema views, execute system stored procedures, or use system functions. You also could query the system tables directly, but it is not advised because they may change in subsequent product versions.

3. You want to reference two tables that reside in different databases on the same server. From the **Inventory** database, you want to reference a table in the **Sales** database. How would you reference the table in the **Sales** database in a query?

   You would reference the table in the Sales database by using a fully qualified name. For example, you could specify Sales.dbo.tablename or Sales..tablename.