MICROSOFT
TRAINING
AND CERTIFICATION

Microsoft® Official Curriculum

Module 4: Managing Security

Contents

Microsoft®

Development Lead: Xandria Eykel
Technical Lead: Rick Byham
Instructional Designers: Cheryl Hoople, Lin Joyner (Content Master Ltd), Marilyn McGill (Independent Contractor), Gordon Ritchie (Content Master Ltd.),
Subject Matter Experts: Karl Dehmer, Mike Galos, Graeme Malcolm (Content Master), Mary Neville (Content Master Ltd), and Carl Rabeler (Shadow Mountain Computers),
Classroom Automation: Lorrin Smith-Bates
Graphic Artist: Kimberly Jackson (Independent Contractor)
Editing Manager: Lynette Skinner
Editor: Wendy Cleary
Copy Editor: Bill Jones (S&T Consulting)
Production Manager: Miracle Davis
Production Coordinator: Jenny Boe
Production Support: Ed Casper (S&T Consulting), Theano Petersen (S&T Consulting)
Test Manager: Sid Benavente
Courseware Testing: Testing Testing 123
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, Database Management: Margo Crandall
Group Manager, Courseware Infrastructure: David Bramble
Group Product Manager, Content Development: Dean Murray
General Manager: Robert Stewart

Instructor Notes

Presentation:
90 Minutes

Labs:
90 Minutes

This module provides students with details on implementing security. It begins with a description of how to set up an authentication mode for a server and how to grant access to Microsoft® Windows® 2000 users and groups and Microsoft SQL Server™ 2000 users. The next section describes how to assign login accounts to users and roles and how to assign permissions to users and roles. The module also discusses managing security by using SQL Server, and managing application security. It concludes by describing how to manage SQL Server security in an enterprise environment.

This module has three labs. In the first lab, students set an authentication mode and grant login account access to SQL Server. In the second lab, students assign login accounts to user accounts and roles, and assign permissions to the user accounts and roles. In the third lab, students create an application role, assign it permissions, and test it. Students also create a view and a stored procedure that provide access to a table without granting users direct access to the table.

After completing this module, students will be able to:

- Implement Windows Authentication Mode and Mixed Mode.
- Assign login accounts to database user accounts and roles.
- Assign permissions to user accounts and roles.
- Manage security within SQL Server.
- Manage application security.
- Manage SQL Server security in an enterprise environment.

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 Microsoft PowerPoint® file 2072A_04.ppt.

Preparation Tasks

To prepare for this module, you should:

- Read all of the materials for this module.
- Complete the labs.
- Complete all demonstrations.
- Practice the presentation.
- Review any relevant white papers located on the Trainer Materials compact disc.

Section and Lab Timing

When you practice the presentation, use the following section and lab timings as guidelines.

Section/Lab	Timing (minutes)
Implementing an Authentication Mode	30
Lab A: Managing Security	30
Assigning Login Accounts to Users and Roles	30
Assigning Permissions to Users and Roles	
Managing Security Within SQL Server	
Lab B: Managing Permissions	30
Managing Application Security	15
Lab C: Managing Application Security	30
Managing SQL Server Security in the Enterprise	15

Prerequisites

You must teach this module after you have already taught module 1, “SQL Server Overview”, in course 2072A, *Administering a Microsoft SQL Server 2000Database*. Module 1 introduces basic security concepts on which this module elaborates.

Demonstrations

This section provides demonstration procedures that will not fit in the margin notes or are not appropriate for the student notes.

Setting Up Login Accounts

► **To prepare for the demonstration**

1. Open SQL Server Enterprise Manager.
2. In the console tree, expand your server, and then expand the Security folder.
3. Right-click **Logins**, and then on the shortcut menu, click **New Login**.
4. Add the Windows 2000 group **customer_mgmt** as a SQL Server login account.
5. Add a Windows 2000 user account (one of the **SQLAdmin*x*** accounts).
6. Add a SQL Server login account (**bob**).
7. Demonstrate how to assign users access to a particular database and a default database.
8. Also point out that you can add users to roles and assign permissions from the same dialog box, but do not demonstrate this at this time.

Assigning Login Accounts to User Accounts and Roles

In this demonstration, you will use SQL Server Enterprise Manager to show students how to assign login accounts to user accounts and roles.

► **To demonstrate assigning login accounts to user accounts and roles**

1. Open SQL Server Enterprise Manager.
2. In the console tree, expand the **Northwind** database, right-click the **Users** folder, and then on the shortcut menu, click **New Database User**.
3. Add **customer_mgmt** as a database user.
4. In the console tree, in the **Users** folder, double-click **customer_mgmt**, and then assign permissions to the **customer_mgmt** account.
5. Right-click **Roles**, and on the shortcut menu, click **New Database Role**.
6. Add a new role called **employees**, and then add **bob** and **customer_mgmt** as members of the role.

Module Strategy

Use the following strategy to present this module:

- Implementing an Authentication Mode

  Be sure that you clarify that a SQL Server authentication mode does not exist—the choices are Windows Authentication Mode or Mixed Mode. A client who connects to a server running in Mixed Mode will be asked to choose an authentication mechanism—Windows Authentication or SQL Server Authentication. Also point out the order of precedence that SQL Server uses when clients connect.

- Assigning Login Accounts to Users and Roles

  Describe how users should work with the default user account **dbo**.

  Point out that roles are an easy way to manage permissions for groups of login accounts. In particular, reiterate that members of fixed server roles manage server-level tasks, and members of fixed database roles manage database-level tasks. Be sure that you point out that the permissions of the **public** role in the **Northwind** database are unusual—**Northwind** is a sample database, so the **public** role has permission to do everything.

- Assigning Permissions to Users and Roles

  Make sure that students understand that denying a permission at any level restricts users from performing actions. Also, be sure that they understand that revoking a permission returns the permission to a neutral state in which the permission can be granted by a role.

  When you demonstrate applying permissions by using SQL Server Enterprise Manager, be sure to point out how to grant and revoke column permissions.

- Managing Security Within SQL Server

  This topic is one of the most important in the module. Make sure that students understand that they should carefully consider each of the points in this topic.

- Managing Application Security

  As you teach this section, draw parallels between the security capabilities that views and stored procedures provide and those that application roles provide.

- Managing SQL Server Security in the Enterprise

  This section is a high level overview of subjects related to security in the enterprise. It is intended to introduce students to some of the security features included in Windows 2000 and Microsoft BackOffice® products.

Customization Information

This section identifies the lab setup configuration changes that occur on student computers during the lab. 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 2072A, *Administering a Microsoft SQL Server 2000 Database*.

Lab Setup

The lab in this module requires that SQL Server 2000 Enterprise Edition has been installed on student computers. To prepare student computers to meet this requirement, perform exercise 1 in lab A, “Installing SQL Server,” in module 2, “Planning to Install SQL Server” of course 2072A, *Administering a Microsoft SQL Server 2000 Database*.

Lab Results

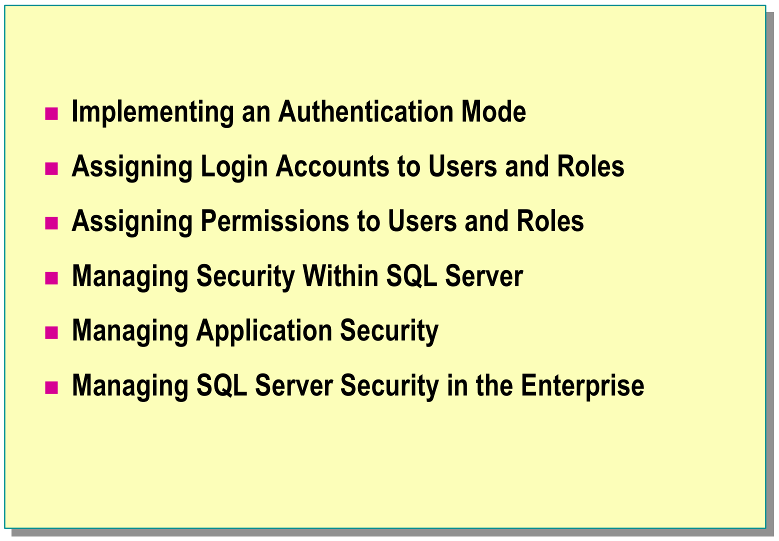
Performing the lab in this module introduces the following configuration change:

- Login accounts and user accounts will be created within SQL Server.
- Permissions for the Public role in the **Northwind** database will be altered.

Overview

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

Lead-in
In this module, you will learn about SQL Server security at both the server and database levels.


- Implementing an Authentication Mode
- Assigning Login Accounts to Users and Roles
- Assigning Permissions to Users and Roles
- Managing Security Within SQL Server
- Managing Application Security
- Managing SQL Server Security in the Enterprise

This module provides details on implementing security. You will learn how to set up an authentication mode and grant access to Microsoft® Windows® 2000 users and groups, and Microsoft SQL Server™ 2000 users. You will then learn how to assign login accounts to users and roles and how to assign permissions to users and roles. Finally, you will learn how to manage security by using SQL Server, how to manage application security, and how to manage security in an enterprise environment.

After completing this module, you will be able to:

- Implement Microsoft Windows Authentication Mode and Mixed Mode.
- Assign login accounts to database user accounts and roles.
- Assign permissions to user accounts and roles.
- Manage security within SQL Server.
- Manage application security.
- Manage SQL Server security in an enterprise environment.

◆ Implementing an Authentication Mode

Topic Objective
To introduce the topic of authentication.

Lead-in
SQL Server security uses two different authentication modes: Windows Authentication Mode and Mixed Mode.

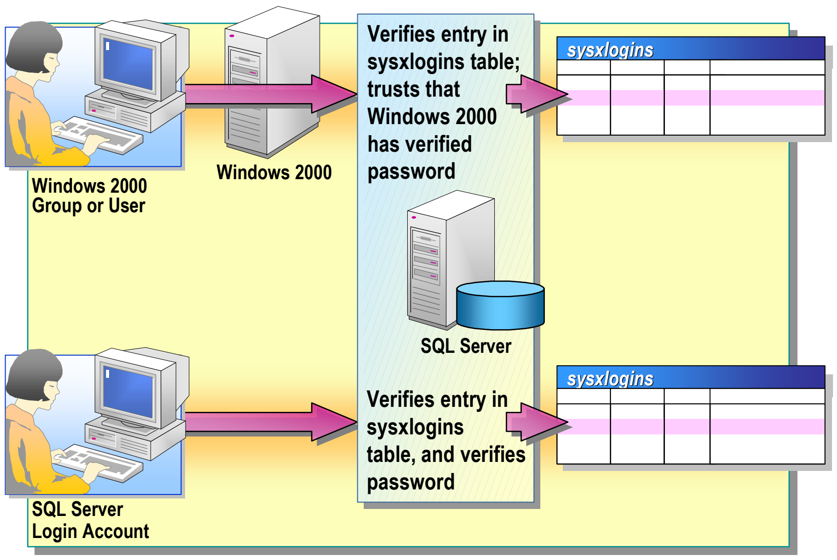
- **Authentication Processing**
- **Choosing an Authentication Mode**
- **Mutual Authentication Using Kerberos**
- **Impersonation and Delegation**
- **Encryption**
- **Steps in Implementing an Authentication Mode**
- **Creating Login Accounts**
- **Setting Up Login Accounts**

You can secure SQL Server 2000 by implementing either Windows Authentication Mode or Mixed Mode. This section describes the process of authentication in each mode, the steps that you must take in implementing authentication, and how to create login accounts.

Authentication Processing

Topic Objective
To introduce how login accounts are processed.

Lead-in
SQL Server can rely on Windows 2000 to authenticate login accounts, or it can authenticate login accounts itself.



Windows 2000 Group or User
Windows 2000
Verifies entry in sysxlogins table; trusts that Windows 2000 has verified password
sysxlogins
SQL Server
SQL Server Login Account
Verifies entry in sysxlogins table, and verifies password
sysxlogins

Delivery Tip
Ask: What can users do after they have been authenticated if their login accounts do not have permissions in any databases?

Answer: They can access the server and some system objects but cannot access any user databases.

SQL Server can rely on Windows 2000 to authenticate login accounts, or it can authenticate login accounts itself.

How SQL Server Processes Login Accounts That Windows 2000 Authenticates

The following describes how SQL Server processes login accounts that Windows 2000 authenticates:

- When a user connects to SQL Server, the client opens a trusted connection to SQL Server, which passes the user's Windows 2000 security credentials to SQL Server.

  Because the client opened a trusted connection, SQL Server knows that Windows 2000 has already validated the login account.

- If SQL Server finds the user's Windows 2000 user account or group account in the list of SQL Server login accounts in the **sysxlogins** system table, it accepts the connection.

  SQL Server does not need to revalidate a password because Windows 2000 has already validated it.

Note SQL Server will not recognize users or groups that you have deleted and recreated in Windows 2000. Windows 2000 accounts are internally identified by using a unique security identifier (SID) that is never reused. You must delete the account from SQL Server and add it again, because SQL Server uses the SID to identify the account.

- If multiple computers running SQL Server participate in a domain or a group of trusted domains, logging on to a single network domain is sufficient to enable access to all computers running SQL Server.

Note The SQL Server command line utilities support options that allow you to connect using a trusted connection.

How SQL Server Processes Login Accounts That It Authenticates

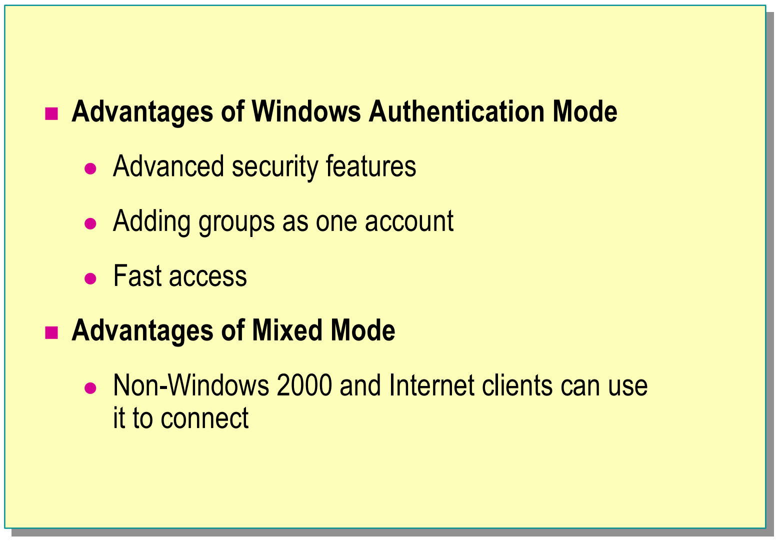
SQL Server takes the following steps to process login accounts that it authenticates:

- When a user connects with a SQL Server login account and password, SQL Server verifies that a login account exists in the **sysxlogins** table and that the specified password matches the previously recorded password.
- If SQL Server does not have a login account set up to accommodate the user, authentication fails and the connection is refused.

Choosing an Authentication Mode

Topic Objective
To compare the advantages of the two security authentication modes.

Lead-in
The security needs of your server and network environments will determine the authentication mode that you use for your SQL Server.



- **Advantages of Windows Authentication Mode**
  - Advanced security features
  - Adding groups as one account
  - Fast access
- **Advantages of Mixed Mode**
  - Non-Windows 2000 and Internet clients can use it to connect

The security needs of your server and network environments will determine the authentication mode that you use for your SQL Server. You can use SQL Server Enterprise Manager to set the authentication mode of your server.

Advantages of Windows Authentication Mode

Use Windows Authentication Mode in network environments in which all clients support trusted connections.

Windows Authentication offers several advantages over SQL Server Authentication by:

- Providing more features, such as secure validation and encryption of passwords, auditing, password expiration, minimum password length, and account lockout after an invalid password.
- Enabling you to add groups of users to SQL Server by adding a single login account.
- Enabling users to access SQL Server quickly, without having to remember another login account and password.

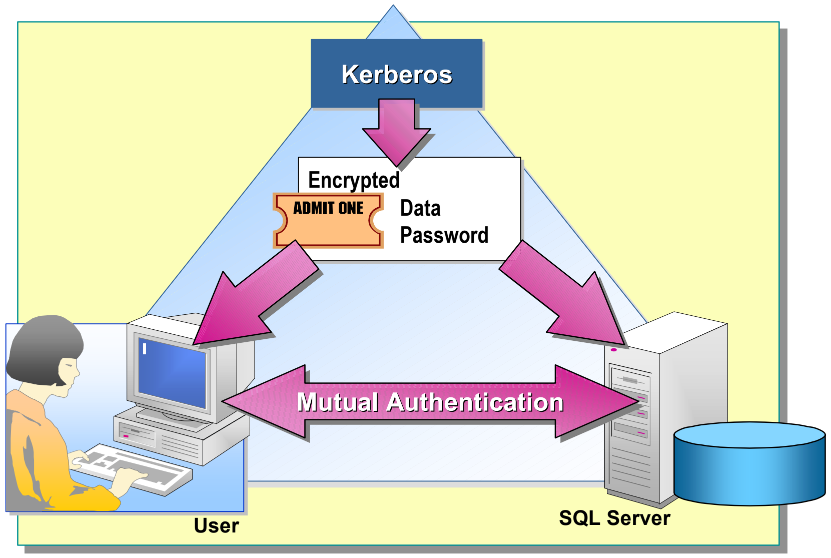
Advantages of Mixed Mode

Mixed Mode, and the SQL Server Authentication mechanism in particular, enables non-Windows 2000 clients, Internet clients, and mixed client groups to connect to SQL Server.

Mutual Authentication Using Kerberos

Topic Objective
To introduce the basic principles of the Kerberos protocol.

Lead-in
Kerberos is the primary security protocol for authentication within Windows 2000.



Kerberos is the primary security protocol for authentication within a Windows 2000 domain. It defines how clients interact with network authentication services.

The Kerberos protocol verifies that both the identity of the user and network services. This dual authentication is known as mutual authentication. SQL Server 2000 uses Kerberos to support mutual authentication between the client and the server.

When you log on, Windows 2000 locates an Active Directory™ directory service server and Kerberos authentication service. The Kerberos service issues a ticket-granting ticket (TGT) containing encrypted data that confirms your identity to the Key Distribution Center (KDC). When you require access to a service, you send this TGT to the KDC, which will issue a session ticket to you for the required service. You then present the session ticket, which confirms your identity, to the service.

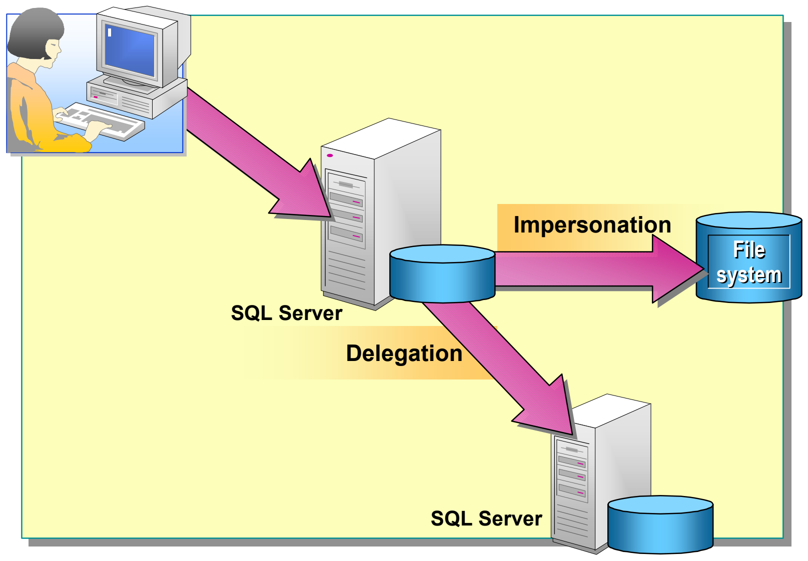
If mutual authentication is required, the server then responds with an encrypted message to identify itself. The session ticket can be reused for access to the same service until it expires, which eliminates the need for the KDC to be constantly issuing repeat tickets. Expiration time is determined by the KDC, but is typically no longer than eight hours, the length of a normal logon session.

Note Successful authentication using Kerberos requires that both the client system and the server running the desired service must run the Windows 2000 operating system and use the TCP/IP sockets network library, and that a Service Principal Name (SPN) must be properly configured for the SQL Server service.

Impersonation and Delegation

Topic Objective
To explain the concepts of impersonation and delegation.

Lead-in
Impersonation and delegation allow SQL Server to use the security credentials of the original client when accessing resources on local or remote servers.



Impersonation and delegation allow SQL Server to use the security credentials of the original client when accessing resources on local or remote servers.

Impersonation

In some circumstances, SQL Server must present a client's identity to resources that it accesses on the client's behalf, for example, to the file system. The server can impersonate the client's security context to allow access checks or authentication to be performed against the client's identity, ensuring that access to resources are neither restricted nor expanded beyond the client's own permissions.

Delegation

SQL Server 2000 also supports delegation, which is the ability to connect to multiple servers and, with each server change, to retain the credentials of the original client. For example, if a user (NWTraders\SQLAdmin20) connects to ServerA, which then connects to ServerB, ServerB knows that the connection security identity is NWTraders\SQLAdmin20.

To use delegation, all servers to which you are connecting must be running Windows 2000 with Kerberos support enabled, and you must be using Active Directory.

Configuring Active Directory for Delegation

You must set the following account options in Active Directory for delegation to work:

- Account is sensitive and cannot be delegated

  You must *not* select this option for the user requesting delegation.

- Account is trusted for delegation

  You must select this option for the service account of SQL Server.

- Computer is trusted for delegation

  You must select this option for the computer running SQL Server.

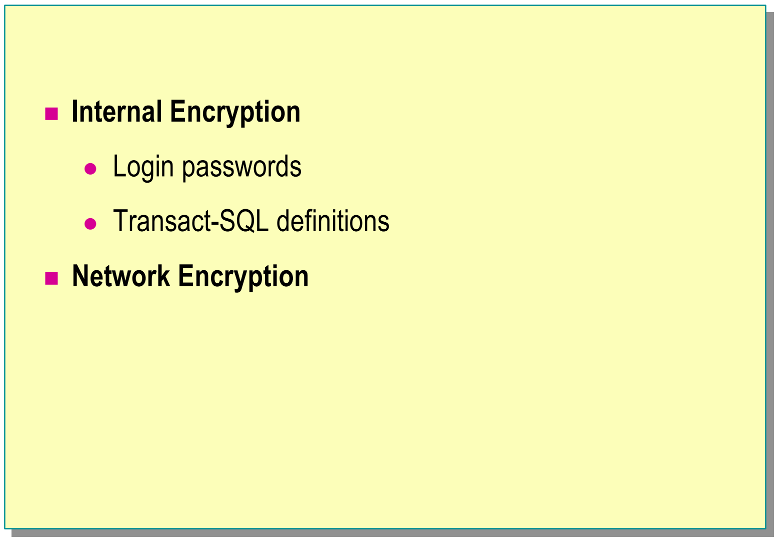
Configuring SQL Server for Delegation

To use security account delegation, SQL Server must have an SPN assigned by the Windows 2000 account domain administrator to the SQL Server service account and be using Transmission Control Protocol/Internet Protocol (TCP/IP). If the SQL Server service is running under the LocalSystem account, a SPN is automatically registered by the SQL Server service at service startup, and de-registered when SQL Server is shut down.

Encryption

Topic Objective
To introduce the encryption facilities available in SQL Server.

Lead-in
Encryption is a method for protecting confidential information by changing data into an unreadable format.



- **Internal Encryption**
  - Login passwords
  - Transact-SQL definitions
- **Network Encryption**

Encryption is a method for protecting confidential information by changing data into an unreadable format. It ensures that data remains secure, even if viewed directly.

Internal Encryption

SQL Server can encrypt:

- Login passwords stored in SQL Server.

  Login passwords stored in system tables are always encrypted. This prevents all users, including system administrators, from viewing any password, including their own.

- Transact-SQL definitions.

  Definitions of stored procedures, triggers, and views stored in the **syscomments** system table can be encrypted.

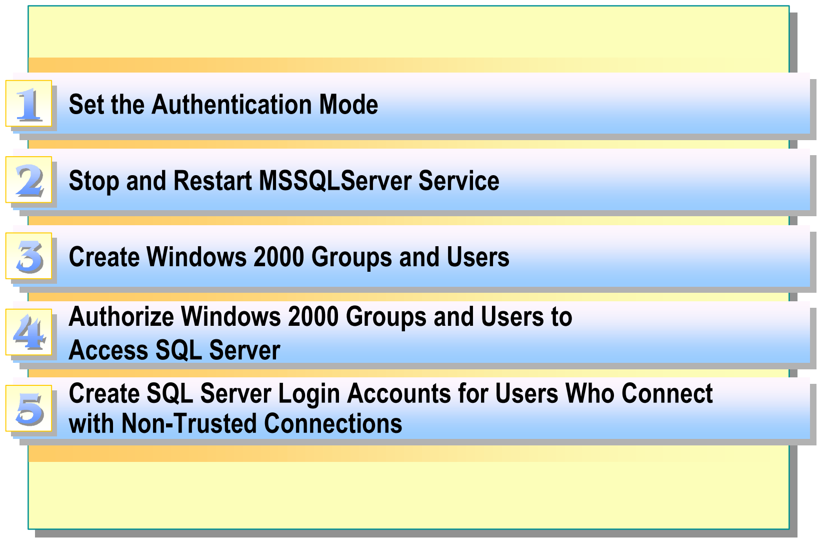
Network Encryption

SQL Server allows data sent between the client and the server to be encrypted. This ensures that any application or user intercepting the data packets on the network cannot view confidential or sensitive data, for example, passwords sent across the network as a user logs into SQL Server, or personnel data containing salary information.

Steps in Implementing an Authentication Mode

Topic Objective
To describe the steps that are necessary to implement authentication.

Lead-in
You must perform the following tasks to implement authentication.


1 Set the Authentication Mode

2 Stop and Restart MSSQLServer Service

3 Create Windows 2000 Groups and Users

4 Authorize Windows 2000 Groups and Users to Access SQL Server

5 Create SQL Server Login Accounts for Users Who Connect with Non-Trusted Connections

You must perform the following tasks from a system administration account to implement authentication. For Windows Authentication Mode, perform Steps 1–4; for Mixed Mode, perform Steps 1–5:

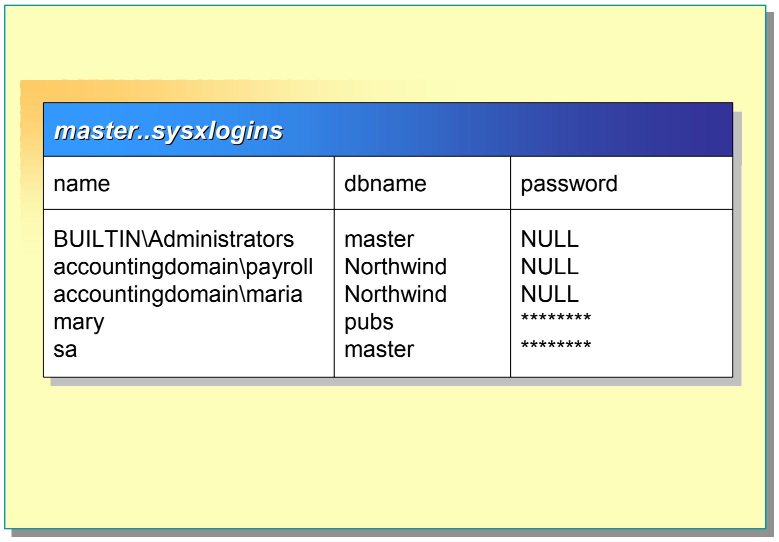
1. Use SQL Server Enterprise Manager to set the authentication mode for SQL Server.
2. Stop and then restart MSSQLServer service for the security option to take effect.
3. Create the Windows 2000 groups and users that are authorized to access SQL Server over trusted connections.

   If you do not have permission to administer Windows 2000 groups and users, have a Windows 2000 administrator perform this task for you.
4. Use SQL Server Enterprise Manager to grant Windows 2000 groups and users access to SQL Server.
5. Use SQL Server Enterprise Manager to create SQL Server login accounts for users to connect with non-trusted connections.

Creating Login Accounts

Topic Objective
To introduce the process of creating login accounts.

Lead-in
You can create login accounts from existing Windows 2000 users and groups, or you can create new SQL Server login accounts.



master..sysxlogins

name	dbname	password
BUILTIN\Administrators	master	NULL
accountingdomain\payroll	Northwind	NULL
accountingdomain\maria	Northwind	NULL
mary	pubs	********
sa	master	********

You can create login accounts from existing Windows 2000 users and groups, or you can create new SQL Server login accounts. You also can use one of the default login accounts. Login accounts are stored in the **master..sysxlogins** system table. When a login account is added to a SQL Server, it often is assigned a default database. Assigning a default database to a login account does not create a user account in that database—it sets the default context for actions that the user takes.

Adding a Windows 2000 Login Account to SQL Server

You can use SQL Server Enterprise Manager or the **sp_grantlogin** system stored procedure to allow a Windows 2000 user or group account to connect to SQL Server. Only system or security administrators can execute **sp_grantlogin**.

For Your Information
Windows 2000 allows longer user names than SQL Server does.

The name specified when creating an account is the name of the Windows 2000 user or group to be added. This name must be qualified with a Windows 2000 domain name. The limit for combined domain and user or group names in SQL Server is 128 Unicode characters.

Consider the following facts and guidelines about adding Windows 2000 login accounts to SQL Server:

- Because SQL Server has a single login account for a Windows 2000 group, no changes to SQL Server are required when membership in a Windows 2000 group changes. This prevents orphaned objects (objects that are owned by a user who no longer exists in SQL Server), as long as you do not delete the group.
- Deleting a Windows 2000 group or a user does not delete that group or user from SQL Server. This prevents orphaned objects.

  When you remove Windows 2000 users or groups, you first should remove them from Windows 2000 in order to disallow network access. You then should remove them from SQL Server.

- Add a login account for a Windows 2000 group account if every member of the group will be connecting to the SQL Server.
- Add a login account for a Windows 2000 user account if the user is not a member of a group that can be granted permission collectively.
- Even though users log in to SQL Server as members of Windows 2000 groups, SQL Server still knows the identities of the users. The **SUSER_SNAME()** function returns the users' domain and login account names when users are members of a Windows 2000 group.
- SQL Server only supports specifying Windows accounts in the form DOMAIN\UserName. SQL Server does not support adding Windows accounts through the use of User Principal Names (UPNs) (names in the form of *someone*@microsoft.com).

The following table lists other system stored procedures that you can use for managing Windows 2000 login accounts. Only system or security administrators can execute these system stored procedures.

System stored procedure	Description
sp_revokelogin	Removes the login account entries for a Windows 2000 user or group from SQL Server
sp_denylogin	Prevents a Windows 2000 user or group from connecting to SQL Server

You can also deny access from login accounts by using **SQL Server Login Properties**.

Adding a SQL Server Login Account

You can use SQL Server Enterprise Manager or the **sp_addlogin** system stored procedure to create a SQL Server login account. Only system or security administrators can execute **sp_addlogin**.

Creating a new SQL Server login account adds a record to the **sysxlogins** table of the **master** database.

SQL Server login accounts and passwords can contain up to 128 characters, including letters, symbols, and digits. However, login accounts cannot:

- Contain a backslash character.
- Be a reserved login account—for example, **sa**.
- Be NULL or an empty string ("").

Users can change their own passwords at any time by using the **sp_password** system stored procedure. System administrators can change any user's password by using **sp_password** with NULL as the old password. Users and system administrators can also change passwords by using **SQL Server Login Properties**.

Default Login Accounts

For Your Information
You can use the BUILTIN prefix with the default Windows 2000 group names only.

SQL Server has two default login accounts: **BUILTIN\Administrators** and **sa**.

BUILTIN\Administrators is provided as the default login account for all Windows 2000 administrators. It has all rights on the SQL Server and in all databases.

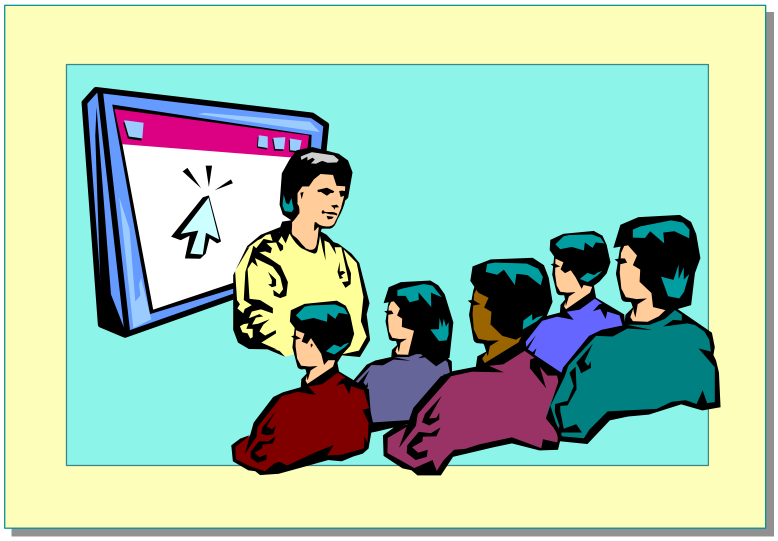
System administrator (**sa**) is a special login account that has all rights on the SQL Server and in all databases. It is provided for backward compatibility and should not be used routinely. This account is only enabled when SQL Server is using Mixed Mode authentication.

Note When using Mixed Mode, ensure that the **sa** password is not blank; otherwise, your SQL Server will not be secure.

Demonstration: Setting Up Login Accounts

Topic Objective
To demonstrate how to create login accounts.

Lead-in
In this demonstration, I will add login accounts to SQL Server by using SQL Server Enterprise Manager.



In this demonstration, you will view how to add login accounts to SQL Server by using SQL Server Enterprise Manager.

Lab A: Managing Security

Topic Objective
To introduce the lab.

Lead-in
In this lab, you will set up Windows Authentication mode and enable Windows 2000 login accounts to access SQL Server.

Explain the lab objectives.

Objectives

After completing this lab, you will be able to:

- Configure Windows Authentication Mode.
- Authorize Windows 2000 groups and users to access SQL Server 2000.
- Revoke and deny access for Windows 2000 groups and users.

For More Information

If you require help in 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 that of 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 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
Configuring Windows Authentication Mode

In this exercise, you will verify that SQL Server is configured to use Windows Authentication to allow trusted connections to access SQL Server.

► To verify that SQL Server is using Windows Authentication

In this procedure, you will use SQL Server Enterprise Manager to verify that Windows Authentication is enabled.

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. Start SQL Server Enterprise Manager.

   You have permission to log in to and administer SQL Server because you are logged as **SQLAdmin*x***, which is a member of the Windows 2000 local group, Administrators. All members of this group are automatically mapped to the SQL Server **sysadmin** role.

3. In the console tree, expand **Microsoft SQL Servers**, and then expand **SQL Server Group**.
4. Right-click your server, and then click **Properties**.
5. Click the **Security** tab.
6. Verify that **Windows only** is selected, and then click **OK**.

Exercise 2
Authorizing Users to Access SQL Server

In this exercise, you will authorize Windows 2000 users and groups to access SQL Server with a trusted connection.

For the purpose of this lab, each user will be allowed to use a trusted connection. You will use the users and groups that you create in this exercise in subsequent exercises in this lab.

The following table lists the Windows 2000 users and groups that have already been created on the NWTraders classroom domain.

Windows 2000 group	Members	User password
customer_mgmt	carl	**password**
	kathy	**password**
	lisa	**password**
	david	**password**
Domain Users	paul	**password**
	max	**password**

► To authorize a Windows 2000 user or group to access SQL Server

In this procedure, you will use SQL Server Enterprise Manager to authorize existing Windows 2000 user and group accounts to use trusted connections to access SQL Server.

1. In SQL Server Enterprise Manager, expand your server, and then expand **Security**.
2. Right-click **Logins**, and then click **New Login**.
3. Use the information in the following table to allow these Windows 2000 accounts to access your SQL Server.

Name	Default database	Database access
NWTraders\paul	**Northwind**	**Northwind**
NWTraders\carl	**Northwind**	**Northwind**
NWTraders\Kathy	**Northwind**	**Northwind**
NWTraders\customer_mgmt	**Northwind**	**Northwind**

► To test the new Windows 2000 login accounts

In this procedure, you will use SQL Query Analyzer to verify the Windows 2000 login accounts that you set up in previous steps.

1. Log off Windows 2000, and log on to the **NWTraders** classroom domain by using the information in the following table.

Option	Value
User name	**david**
Password	**password**

2. Open SQL Query Analyzer and, if requested, log in to the (local) server with Windows Authentication.

   You have permission to log in to and administer SQL Server because you are logged as **SQLAdmin*x***, which is a member of the Windows 2000 local group, Administrators. All members of this group are automatically mapped to the SQL Server **sysadmin** role.

   How did **david** connect to the database when his Windows 2000 login account is not authorized to use SQL Server?

   david is a member of the customer_mgmt group in Windows 2000, which has been authorized to use SQL Server.

   ____

   ____

3. Log off Windows 2000.
4. Log on to the **NWTraders** classroom domain by using the information in the following table.

Option	Value
User name	**max**
Password	**password**

5. Open SQL Query Analyzer and, if requested, log in to the (local) server with Windows Authentication.
6. You have permission to log in to and administer SQL Server because you are logged as **SQLAdmin*x***, which is a member of the Windows 2000 local group, Administrators. All members of this group are automatically mapped to the SQL Server **sysadmin** role.

   What happens and why?

   max is denied access to SQL Server. He is a valid Windows 2000 user, but he has not been authorized to use SQL Server and does not belong to a Windows 2000 group that has been authorized to access SQL Server.

   ____

   ____

7. Log off Windows 2000.

Exercise 3
Revoking Access from, and Denying Access to, Windows 2000 Users or Groups

In this exercise, you will revoke access from and deny access to Windows 2000 users and groups.

► To revoke access from a Windows 2000 user or group

In this procedure, you will revoke access from the NWTraders\customer_mgmt group.

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. Start SQL Server Enterprise Manager.
3. Expand **Microsoft SQL Servers**, expand **SQL Server Group**, and then expand your server.
4. Expand **Security**, and then click **Logins**.
5. In the **Details** pane, right-click **NWTraders\customer_mgmt**, and then click **Delete**.
6. Confirm that you want to delete the login account.

   Can Windows 2000 users who are members of this group access SQL Server after the group has been removed from SQL Server?

   Yes. Windows 2000 users who have been granted access to SQL Server through their Windows 2000 user accounts (for example, kathy and carl) or who belong to another group that has been granted access can still access SQL Server.

   ____

   ____

► **To deny access to a Windows 2000 group or user**

In this procedure, you will deny access to the NWTraders\paul account.

1. Start SQL Server Enterprise Manager.
2. Expand **Microsoft SQL Servers**, expand **SQL Server Group**, and then expand your server.
3. Expand **Security**, and then click **Logins**.
4. In the **Details** pane, right-click **NWTraders\paul**, and then click **Properties**.
5. In **Security access**, select **Deny access**, and then click **OK**.

   Can paul access SQL Server?

   No. His Windows 2000 login account has specifically been denied access to SQL Server.

   ____

   ____

   Now assume that NWTraders\paul is a member of another Windows 2000 group that has been granted access to SQL Server. Can paul access SQL Server?

   No. His Windows 2000 login account has specifically been denied access to SQL Server. Even membership in another group that has been granted access will not enable paul to access SQL Server.

   ____

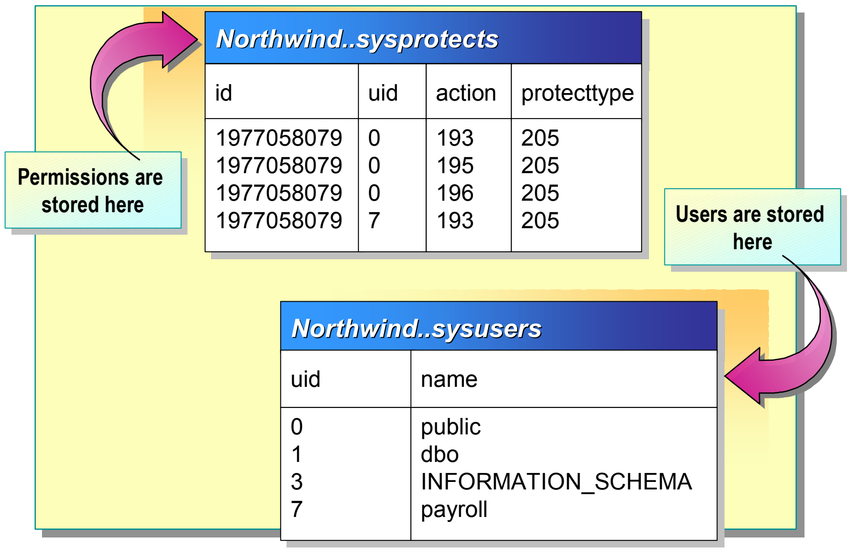
   ____

6. Log off Windows 2000.

◆ Assigning Login Accounts to Users and Roles

Topic Objective
To show the relationship between system tables.

Lead-in
After you add login accounts to SQL Server, you can map then to user accounts or roles in each database to which users need access.



Northwind..sysprotects

id	uid	action	protecttype
1977058079	0	193	205
1977058079	0	195	205
1977058079	0	196	205
1977058079	7	193	205

Northwind..sysusers

uid	name
0	public
1	dbo
3	INFORMATION_SCHEMA
7	payroll

For Your Information
The column values for the **Northwind..sysprotects** table are shown below.
ID:
1977058079 = **Employees** table
Action:
193 = SELECT
195 = INSERT
196 = DELETE
197 = UPDATE
protecttype:
204 = GRANT_W_GRANT
205 = GRANT
206 = REVOKE

Delivery Tip
Go through this example with students, and then ask students to identify how a member of the Windows 2000 global group, **payroll**, can access the **Employees** table.

After you add login accounts to SQL Server, you can map them to user accounts or roles in each database to which users need access.

The **sysusers** system table in a database contains one row for each Windows 2000 user, Windows 2000 group, SQL Server user, or role in the database. Permissions are applied to entries in the **sysusers** table and stored in the **sysprotects** table of the current database.

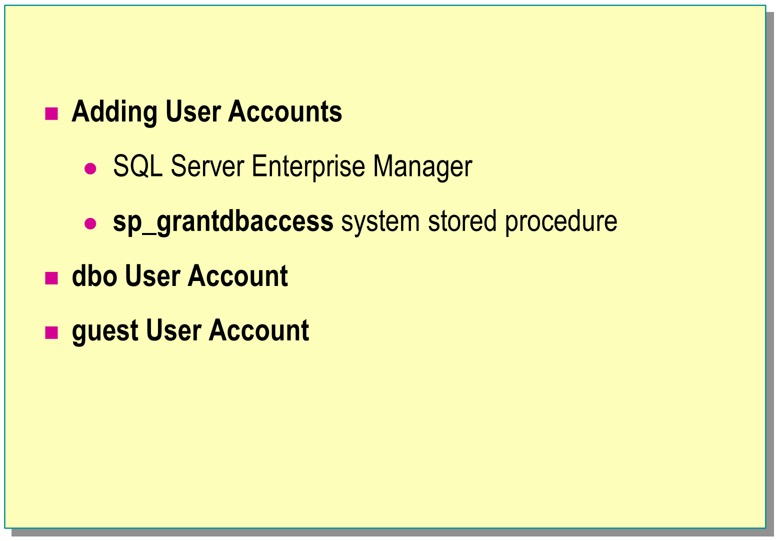
For example, in the illustration above, members of the Windows 2000 **payroll** group have been granted access to the **Northwind** database. After connecting to SQL Server, the user has:

- A validated connection to SQL Server, because the user's Windows 2000 user account is in the Windows 2000 global group, **payroll**.
- No individual permissions in the **Northwind** database—only the **payroll** global group has permissions.

Assigning Login Accounts to User Accounts

Topic Objective
To introduce database user accounts.

Lead-in
To access a database, a login account can use either a default user account or an assigned user account.



- **Adding User Accounts**
  - SQL Server Enterprise Manager
  - **sp_grantdbaccess** system stored procedure
- **dbo User Account**
- **guest User Account**

To access a database, a login account can use either an assigned user account or one of the default user accounts.

Adding User Accounts

To add a user account to a database, you can use SQL Server Enterprise Manager or execute the **sp_grantdbaccess** system stored procedure. You can also add user accounts to databases when creating login accounts. Only database owners or database access administrators can execute **sp_grantdbaccess**.

In the console tree in SQL Server Enterprise Manager, each database has a Users folder. This displays the current list of users for the database, and allows you to add users, remove users and customize their properties.

The following table lists other system stored procedures that you can use for managing database access.

System stored procedure	Description
sp_revokedbaccess	Removes a security account from the current database
sp_change_users_login	Changes the relationship between a SQL Server login account and a SQL Server user in the current database

dbo User Account

The **sa** login and members of the **System Administrators** (**sysadmin**) role are mapped to a special user account inside all databases called **dbo**. Any object that a system administrator creates automatically belongs to **dbo**. The **dbo** user is a default account and cannot be deleted.

guest User Account

The **guest** user account allows logins without user accounts to access a database. Login accounts assume the identity of the **guest** user when both of the following conditions are met:

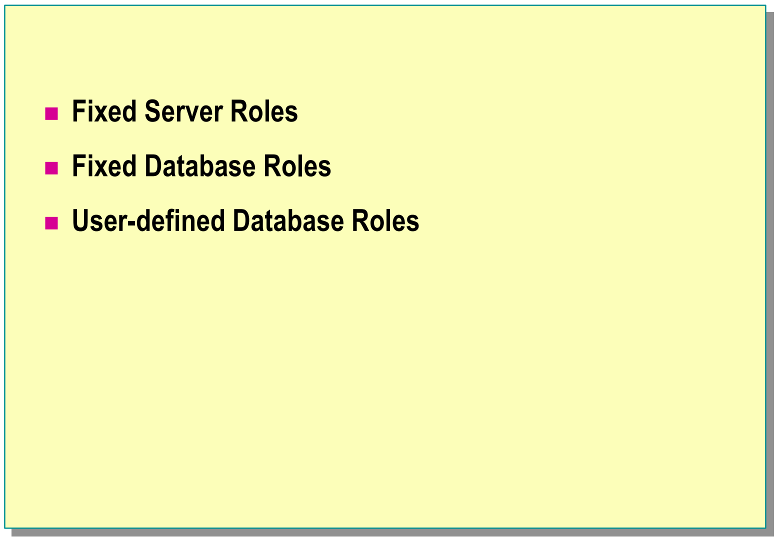
- The login account has access to SQL Server but does not have access to the database through its own user account.
- The database contains a **guest** user account.

Permissions can be applied to the **guest** user as if the **guest** user were any other user account. You can delete and add the **guest** user to any database except the **master** and **tempdb** databases.

◆ Assigning Login Accounts to Roles

Topic Objective
To introduce SQL Server roles.

Lead-in
Roles provide a means of assembling users into a single unit to which permissions can be applied.



- **Fixed Server Roles**
- **Fixed Database Roles**
- **User-defined Database Roles**

Roles provide a means of assembling users into a single unit to which permissions can be applied.

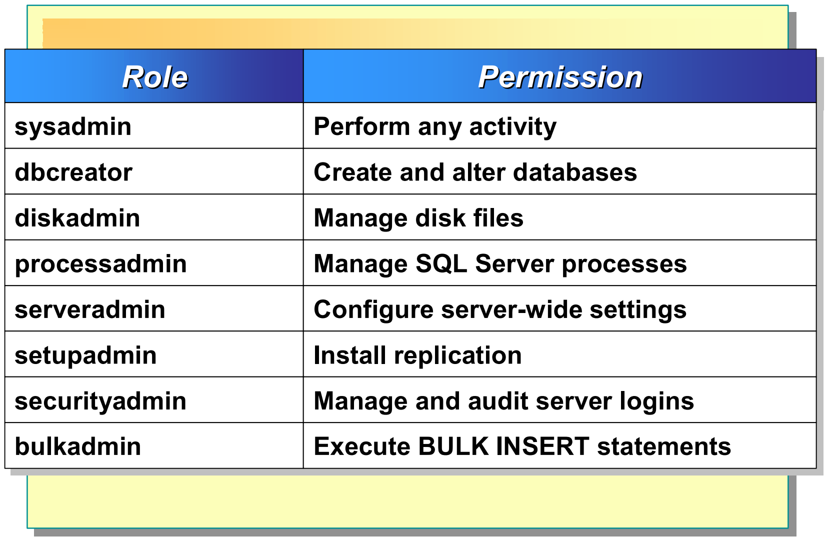
SQL Server provides predefined server and database roles for common administrative functions so that you can easily grant a selection of administrative permissions to a particular user.

You also can create your own database roles to represent tasks that a class of employees in your organization performs. As an employee moves into a given position, you simply add the employee as a member of the role; remove the employee from the role when the employee moves out of the position. You do not have to grant and revoke permissions repeatedly as employees commence or end various positions. If the function of a position changes, it is easy to change the permissions for the role and have the changes automatically apply to all members of the role.

Fixed Server Roles

Topic Objective
To introduce fixed server roles.

Lead-in
Fixed server roles provide groupings of administrative privileges at the server level.



Role	Permission
sysadmin	Perform any activity
dbcreator	Create and alter databases
diskadmin	Manage disk files
processadmin	Manage SQL Server processes
serveradmin	Configure server-wide settings
setupadmin	Install replication
securityadmin	Manage and audit server logins
bulkadmin	Execute BULK INSERT statements

Delivery Tip
Point out that the **sysadmin** role is the equivalent of the **sa** login account.

Members of this role can perform any activity on the server.

Fixed server roles provide groupings of administrative privileges at the server level. They are managed independently of user databases at the server level and are stored in the **master..sysxlogins** system table.

Assigning a Login Account to a Fixed Server Role

You can use **SQL Server Login Properties** in SQL Server Enterprise Manager, or the **sp_addsrvrolemember** system stored procedure, to add a login account as a member of a fixed server role. Only members of fixed server roles can execute the **sp_addsrvrolemember** system stored procedure.

When you add a login account to a server role, the corresponding row for the login account in the **sysxlogins** table is updated to indicate that the login account is a member of the role and has permissions that are associated with the server role.

Consider the following facts about assigning login accounts to fixed server roles:

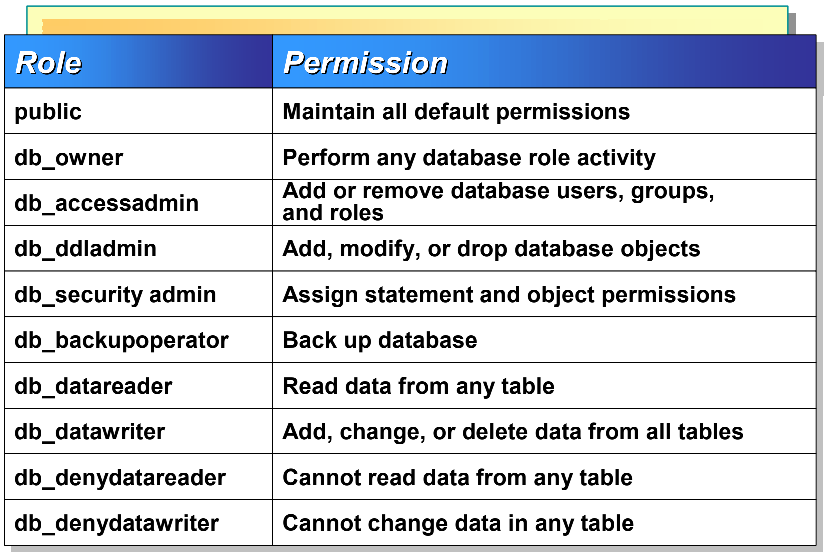
- You cannot add, modify, or remove fixed server roles.
- Any member of a fixed server role can add other login accounts to that role.
- The **sp_addsrvrolemember** system stored procedure cannot be executed within a user-defined transaction.

You also can use the **sp_dropsrvrolemember** system stored procedure to remove a member from a fixed server role.

Fixed Database Roles

Topic Objective
To introduce fixed database roles.

Lead-in
Fixed database roles provide groupings of administrative privileges at the database level.



Role	Permission
public	Maintain all default permissions
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_security admin	Assign statement and object permissions
db_backupoperator	Back up database
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

Delivery Tip
Point out that the **public** role in the **Northwind** database has been granted all permissions. Security is set this way only because **Northwind** is a sample database.

Fixed database roles provide groupings of administrative privileges at the database level. Fixed database roles are stored in the **sysusers** system table of each database.

The public Role

The **public** role is a special database role to which every database user belongs and cannot be removed. The **public** role:

- Maintains all default permissions for users in a database.
- Cannot have users, groups, or roles assigned to it because users, groups, and roles already belong by default.
- Is contained in every database, including **master**, **msdb**, **tempdb**, **model**, and all user databases.
- Cannot be deleted.

Without appropriate permissions, a user can connect to SQL Server but is only able to perform limited tasks. Without permissions, a user possesses any permissions that are granted to the **public** role and can:

- Execute statements that do not require permissions, such as the PRINT statement.
- View system table information and execute certain system stored procedures to retrieve information from the **master** database and user databases to which they have access.
- Gain access to any database with a **guest** account.

Assigning a Security Account to a Fixed Database Role

Use SQL Server Enterprise Manager or the **sp_addrolemember** system stored procedure to add a security account as a member of a fixed database role. Only members of the **db_owner** role can execute the **sp_addrolemember** system stored procedure for any role in the database.

Consider the following facts when you assign security accounts to a fixed database role:

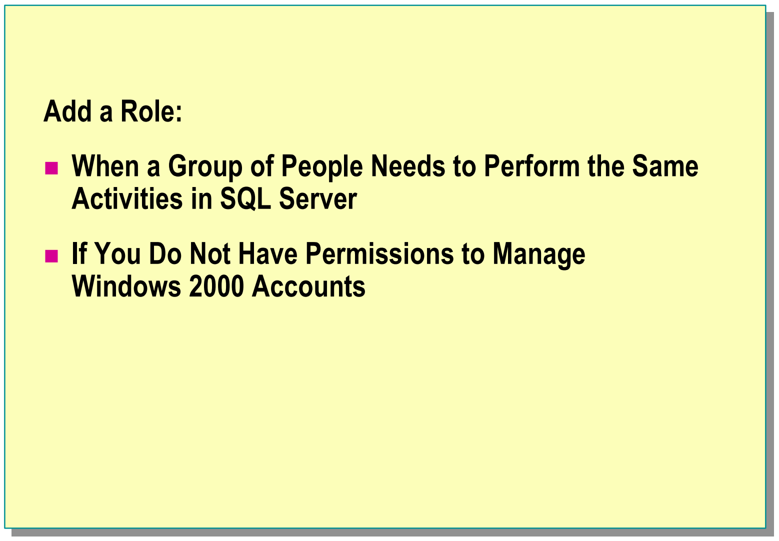
- Fixed database roles cannot be added, modified, or removed.
- Any member of a fixed database role can add other login accounts to that role.

You also can use the **sp_droprolemember** system stored procedure to remove a security account from a role.

User-defined Database Roles

Topic Objective
To introduce user-defined database roles.

Lead-in
Creating a user-defined database role enables you to create a group of users with a set of common permissions.



Add a Role:

- When a Group of People Needs to Perform the Same Activities in SQL Server
- If You Do Not Have Permissions to Manage Windows 2000 Accounts

Creating a user-defined database role enables you to create a group of users with a set of common permissions. Add a user-defined role to the database:

- When a group of people needs to perform a specified set of activities in SQL Server, and no applicable Windows 2000 group exists.
- If you do not have permissions to manage Windows 2000 user accounts.

For example, a company may form a new Charity Event committee that includes employees from different departments at several different levels. These employees need access to a special project table in the database. A Windows 2000 group does not exist that includes only these employees, and there is no other reason to create one in Windows 2000. You could create a user-defined role, **CharityEvent**, for this project and then add individual Windows 2000 user accounts to the role. When permissions are applied, the individual user accounts in the role gain table access.

Creating a User-defined Database Role

Use SQL Server Enterprise Manager or the **sp_addrole** system stored procedure to create a new database role. An entry is added to the **sysusers** table of the current database for each user-defined role. Only members of the **db_securityadmin** or **db_owner** role can execute **sp_addrole**.

Consider the following fact when you create a database role:

- When you apply permissions to the role, each member of the role gains the effects of the permission as if the permission were applied directly to the member's own account.

Assigning a Security Account to a User-defined Database Role

After you create a role, use SQL Server Enterprise Manager or the **sp_addrolemember** system stored procedure to add users or roles as members of the role. Only members of the **sysadmin** fixed server role, or the **db_securityadmin** and **db_owner** fixed database roles, or the role owner can execute **sp_addrolemember** to add a member to a user-defined database role.

Consider the following facts when you assign security accounts to a user-defined database role:

- When you add a security account to a role, any permissions applied to the role are applied to the new member.
- When you add a SQL Server role as a member of another SQL Server role, it is not possible to create recursive roles. Therefore, **security_account** could not be added as a member of **role** if **role** were already a member of **security_account**.

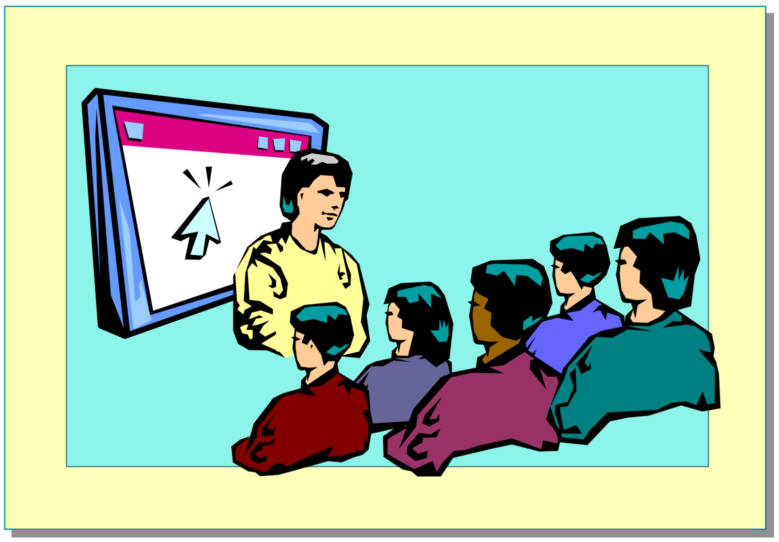
The following table lists additional system stored procedures that you can use for managing database roles.

System stored procedure	Description
sp_droprole	Drops a SQL Server role from the current database
sp_droprolemember	Drops a security account from a SQL Server role

Demonstration: Assigning Login Accounts to User Accounts and Roles

Topic Objective
To demonstrate how to assign login accounts to user accounts and roles.

Lead-in
In this demonstration, I will assign login accounts to user accounts and roles.

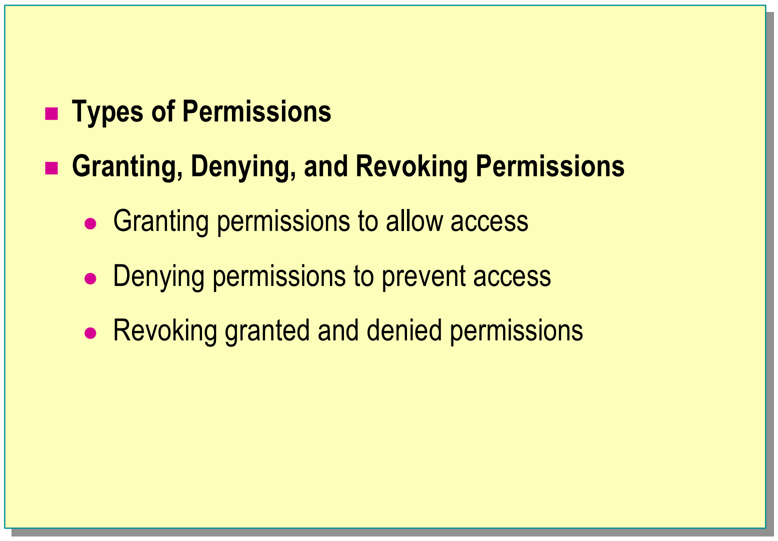


In this demonstration, you will view how to assign login accounts to user accounts and roles by using SQL Server Enterprise Manager.

◆ Assigning Permissions to Users and Roles

Topic Objective
To introduce the general concept of permissions.

Lead-in
After security accounts have been established for a database, you must assign permissions to enforce database security.



- **Types of Permissions**
- **Granting, Denying, and Revoking Permissions**
  - Granting permissions to allow access
  - Denying permissions to prevent access
  - Revoking granted and denied permissions

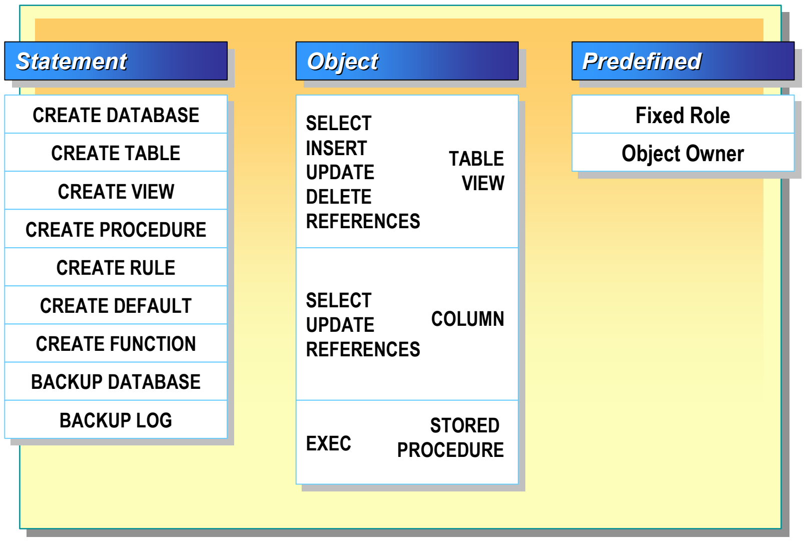
After you have assigned login accounts to user accounts and roles, you must assign permissions to enforce database security.

Permissions specify which database objects users are authorized to use and what the users can do with those objects. The permissions that a user has within a database are dependent on the permissions of the user account and the roles of which the user is a member. It is important to plan the permissions that you grant to each user or group. Each database has its own independent permissions system.

Types of Permissions

Topic Objective
To compare object, statement, and predefined permissions.

Lead-in
There are three types of permissions in SQL Server: statement, object, and predefined.



There are three types of permissions in SQL Server: statement, object, and predefined.

Statement Permissions

Activities that involve creating a database or items in a database require a class of permissions called statement permissions. Statement permissions give users the privilege of issuing certain Transact-SQL statements. Statement permissions, such as CREATE DATABASE, are applied to the statement itself, rather than to a specific item that is defined in the database. Only members of the **sysadmin**, **db_owner**, or **db_securityadmin** role can grant statement permissions.

Object Permissions

Activities that involve working with data or executing procedures require a class of permissions known as object permissions.

Table and View Permissions

Object permissions for tables and views control users' abilities to execute the SELECT, INSERT, UPDATE, and DELETE statements against the table or view.

Note If you want users to be able to use WHERE clauses in UPDATE statements, you must grant them the ability to execute SELECT as well as UPDATE statements.

Column Permissions

SELECT, UPDATE, and REFERENCES permissions can be applied selectively to individual columns.

When a user adds a row to a table with a FOREIGN KEY constraint or changes data in a column with a FOREIGN KEY constraint, SQL Server must validate the data in the column with the data that is referenced in the FOREIGN KEY constraint. If the user does not have SELECT permissions on the referenced column or table, the REFERENCES permission for the column must be granted to the user.

Stored Procedure Permissions

The EXECUTE permission is the only object permission for a stored procedure.

Predefined Permissions

Only members of fixed roles or owners of database objects can perform certain activities. Permissions to perform these activities are called predefined or implicit permissions.

Fixed Role Permissions

Fixed roles have implicit administrative permissions. For example, a user who is added as a member of the **sysadmin** role automatically inherits full permissions to do or read anything in a SQL Server installation. The **sysadmin** role has permissions that cannot be changed, as well as implied permissions that cannot be applied to other user accounts, such as the ability to configure the SQL Server installation.

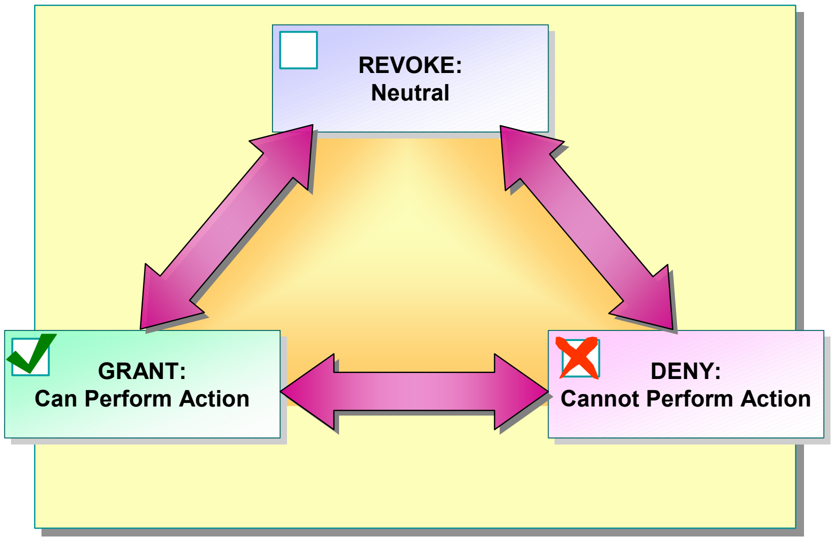
Object Owner Permissions

Object owners also have implied permissions that allow them to perform all activities with objects that they own. For example, a user who is a table owner, or a member of a group that is designated as the table owner, can perform any activity that is related to the table. The user can view, add, or delete data; alter the table definition; and control the permissions that allow other users to work with the table.

◆ Granting, Denying, and Revoking Permissions

Topic Objective
To introduce the concepts of granting, denying, and revoking permissions.

Lead-in
Permissions for a user or role can be granted, denied, or revoked.



Delivery Tip
Point out that the DENY statement prevents users from performing actions. It overrides a permission, whether the permission was granted directly to a user or to a role to which the user belongs.

Permissions for a user or role can be in one of three states: granted, denied, or revoked. Permissions that have not been granted or denied to a user are neutral—as though they had been revoked. Permissions are stored as entries in the **sysprotects** system table in each database. The following table describes the three states of a permission.

Statement	State of entry in sysprotects table	Description
GRANT	Positive	Can perform action
DENY	Negative	Cannot perform action and cannot be overridden by role membership
REVOKE	None	Cannot perform action but can be overridden by role membership

Granted permissions are cumulative—users can perform all of the actions that they have been granted individually and all of the actions that any roles to which they belong have been granted.

The DENY statement prevents users from performing actions. It overrides permission to a role to which the user belongs—whether the permission was granted to a user directly or indirectly.

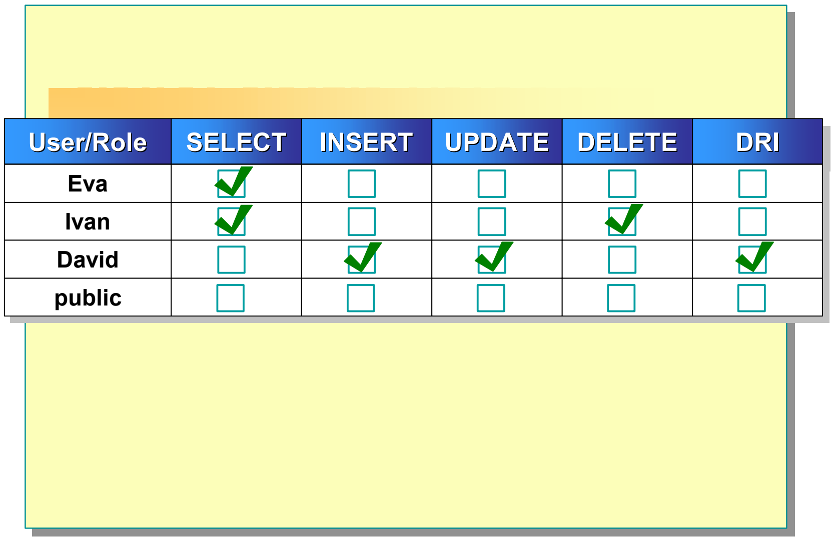
Users have permission to perform an action if both of the following conditions apply:

- They have been granted the permission directly or they belong to a role that has been granted the permission.
- The permission has not been denied to the user directly or to one of the roles to which the user belongs.

Granting Permissions to Allow Access

Topic Objective
To describe how to grant permissions.

Lead-in
You can grant permissions to security accounts and allow the accounts to perform activities or work with data in the database.



User/Role	SELECT	INSERT	UPDATE	DELETE	DRI
Eva	✓	☐	☐	☐	☐
Ivan	✓	☐	☐	✓	☐
David	☐	✓	✓	☐	✓
public	☐	☐	☐	☐	☐

Delivery Tip
Demonstrate how to grant table, view, stored procedure, and column permissions in SQL Server Enterprise Manager.

You grant permissions to security accounts to allow them to perform activities or work with data in a database.

Consider the following facts when you grant permissions:

- You can grant permissions in the current database only.
- The right to grant permissions defaults to members of the **sysadmin**, **db_owner**, and **db_securityadmin** roles and to object owners.

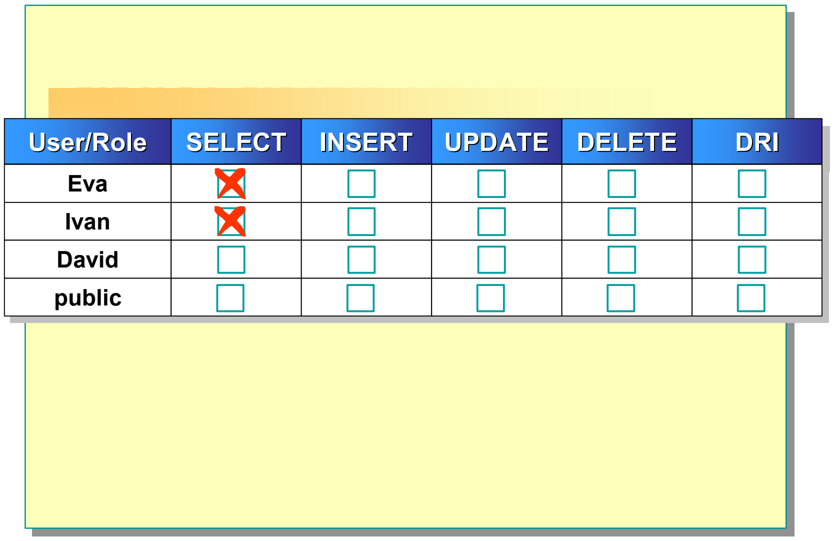
You can grant permissions by using SQL Server Enterprise Manager or the GRANT statement.

The permissions available vary depending on the object selected. For example, a stored procedure has EXECUTE permissions; a table or view has SELECT, INSERT, UPDATE, DELETE, and referential integrity permissions; and individual columns within a table or view have SELECT and UPDATE permissions.

Denying Permissions to Prevent Access

Topic Objective
To describe how to deny permissions.

Lead-in
You occasionally may want to limit the permissions of a certain user or role by denying permissions to that security account.



User/Role	SELECT	INSERT	UPDATE	DELETE	DRI
Eva	❌	☐	☐	☐	☐
Ivan	❌	☐	☐	☐	☐
David	☐	☐	☐	☐	☐
public	☐	☐	☐	☐	☐

Delivery Tip
Demonstrate how to deny table, view, stored procedure, and column permissions in SQL Server Enterprise Manager.

You occasionally may want to limit the permissions of a certain user or role by denying permissions to that security account. Denying permissions on a security account:

- Removes the permissions that are previously granted to the user or role.
- Deactivates permissions that are inherited from another role.
- Ensures that a user or role does not inherit permissions from any other role in the future.

Consider the following facts when you deny permissions:

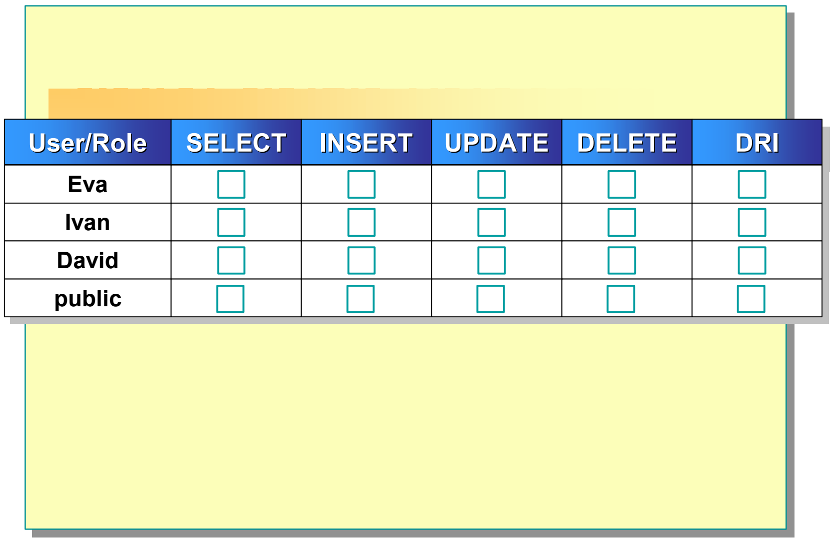
- You can deny permissions in the current database only.
- Permission to deny permissions defaults to members of the **sysadmin**, **db_owner**, and **db_securityadmin** roles and to object owners.

You can use SQL Server Enterprise Manager or the DENY statement to deny permissions.

Revoking Granted and Denied Permissions

Topic Objective
To describe how to revoke permissions.

Lead-in
You can deactivate a granted or denied permission by revoking it.



User/Role	SELECT	INSERT	UPDATE	DELETE	DRI
Eva	☐	☐	☐	☐	☐
Ivan	☐	☐	☐	☐	☐
David	☐	☐	☐	☐	☐
public	☐	☐	☐	☐	☐

Delivery Tip
Demonstrate how to revoke table, view, stored procedure, and column permissions in SQL Server Enterprise Manager.

You can deactivate a granted or denied permission by revoking it. Revoking is similar to denying permissions in that both actions remove a granted permission. The difference is that while revoking a permission removes a granted permission, it does not prevent the user or role from inheriting that permission in the future.

You also can remove a previously denied permission by revoking the DENY statement for the permission.

Consider the following facts when you revoke permissions:

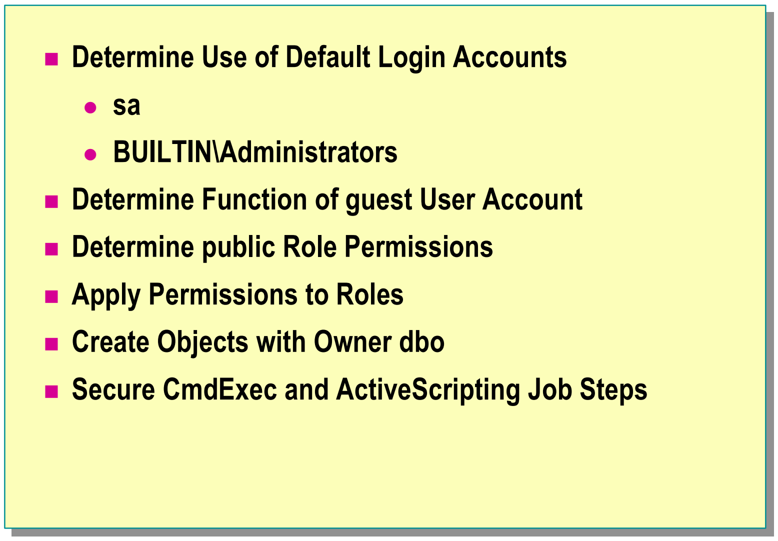
- You can revoke permissions in the current database only.
- Revoking a permission removes the entries in the **syspermissions** system table that were created by granting and denying the permission.
- Permission to revoke permissions defaults to members of the **sysadmin**, **db_owner**, and **db_securityadmin** roles and to object owners.

You can use SQL Server Enterprise Manager or the REVOKE statement to remove a previously granted or denied permission.

Managing Security Within SQL Server

Topic Objective
To introduce topics for consideration in managing security.

Lead-in
You should consider the following recommendations when you plan security.



- Determine Use of Default Login Accounts
  - sa
  - BUILTIN\Administrators
- Determine Function of guest User Account
- Determine public Role Permissions
- Apply Permissions to Roles
- Create Objects with Owner dbo
- Secure CmdExec and ActiveScripting Job Steps

You should consider the following recommendations when you plan security.

For Your Information
The accounts that students use during the labs are members of the **sysadmin** role.

Determine Use of Default Login Accounts

Determine if you will use the default login accounts, and if so, how you will use them.

sa Login Account

System Administrator (**sa**) is a special login provided for backward compatibility. By default, it is assigned to the **sysadmin** fixed server role and cannot be removed. Although **sa** is a built-in administrator login account, it should not be used routinely.

Delivery Tip
Another method of restricting the **BUILTIN\Administrators** login account is to remove domain administrators from the Windows 2000 local group **Administrators**.

BUILTIN\Administrators Login Account

System administrators should be members of the **sysadmin** fixed server role. The Windows 2000 local group **Administrators** is automatically mapped to the SQL Server **BUILTIN\Administrators** login account, which is a member of the **sysadmin** role.

If you do not want all Windows 2000 administrators in your organization to have complete access to your SQL Server, you can remove the **BUILTIN\Administrators** login account or remove the login account from the **sysadmin** role. You can replace the login account with a more appropriate user and assign the new account to the **sysadmin** role. If you replace the login account with a Windows 2000 group, any Windows 2000 administrator will be able to add his or her own user account to that Windows 2000 group.

Determine Function of guest User Account

The **guest** user account allows a login account without a user account to access a database. You should decide whether your databases will have a **guest** account and, if so, what permissions the **guest** account should have in your databases.

Determine public Role Permissions

The **public** role is a special database role to which every database user belongs. It controls the permissions that all users have by default in each database. You should decide which permissions the **public** role will have in each database—by default the **public** role has no permissions.

Apply Permissions to Roles

When applying permissions to roles, follow these guidelines:

- Create user-defined roles for your application. For example, you may have a role for the payroll department, another for the human resources department, and another for the sales department.
- Apply permissions to the user-defined roles that you create for your application. To simplify permissions management, you should avoid applying permissions directly to an individual user.
- Add members to the user-defined roles that you create. Use Windows 2000 groups whenever possible and then add roles or individual users.

Create Objects with Owner dbo

It is very important to determine which users and roles can create objects in a database. In general, it is recommended that only the **sysadmin**, **db_owner**, and **db_ddladmin** fixed database roles be used to create database objects.

It is further recommended that all objects be defined with the **dbo** user specified as the object owner. Defining objects with **dbo** as the owner enables any user in the database to refer to the object without including the owner name. Any object that is created from the **sysadmin** role has **dbo** as the owner. From any other role, always specify the **dbo** user as the owner name when you create the object; otherwise, the object will be created with your user name as the object owner.

Changing Object Owners

If objects were not created with the **dbo** user as the object owner, you can change the object owner by using the **sp_changeobjectowner** system stored procedure.

Syntax

sp_changeobjectowner *object*, *owner*

Consider the following facts about changing database object owners:

- Only members of the **db_owner** and **db_ddladmin** fixed database roles and members of the **securityadmin** server role can change database object owners.
- Changing the owner of a database object requires that you directly update all scripts and batch files that reference that object to include the new ownership information. SQL Server cannot perform this update automatically.

Secure CmdExec and ActiveScripting Job Steps

SQL Server jobs can include CmdExec and ActiveScripting steps. These allow the job to run steps that execute scripts, compiled applications, and batch files. When a user who is not a member of the **sysadmin** role executes these jobs, they are run in the context of the Windows 2000 user account associated with the SQL Server Agent service. This may result in users having the ability to execute functions in the Windows environment in which their Windows accounts are denied.

To prevent this from happening, you can configure the SQL Server Agent service to allow only users with **sysadmin** privileges to execute job steps by using CmdExec and ActiveScripting.

Lab B: Managing Permissions

Topic Objective
To introduce the lab.

Lead-in
In this lab, you will map login accounts to users and roles, create roles, and manage permissions.

Explain the lab objectives.

Objectives

After completing this lab, you will be able to:

- Assign security accounts to server and database roles.
- Create database roles.
- Assign statement and object permissions.

For More Information

If you require help in 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 that of 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 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
Creating Database Roles

In this exercise, you will create user-defined database roles and add users to database roles in the **Northwind** database. You will assign permissions to these roles later in this lab.

► To create a role and add members

In this procedure, you will use SQL Server Enterprise Manager to create a new database role and add members to the **Northwind** database. For now, do not set any permissions. You will use the information in the following table to create a role for users who maintain customer information stored in the **Customer** table.

Role name	Role members
cust_mgmt	**carl**
	kathy

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. Start SQL Server Enterprise Manager.

   You have permission to log in to and administer SQL Server because you are logged as **SQLAdmin*x***, which is a member of the Windows 2000 local group, Administrators. All members of this group are automatically mapped to the SQL Server **sysadmin** role.
3. In the console tree, expand **Microsoft SQL Servers**, and then expand **SQL Server Group**.
4. Expand your server, expand **Databases**, and then expand the **Northwind** database.
5. Right-click **Roles**, and then click **New Database Role**.
6. Under **Name**, type **cust_mgmt**
7. Verify that **Standard role** is selected, click **Add**, and then select **carl** and **kathy** from the list.
8. Click **OK** to close the **Add Role Members** dialog box, and then click **OK** to create the new role.

Exercise 2
Assigning Statement Permissions

In this exercise, you will use SQL Server Enterprise Manager to assign statement permissions to SQL Server security accounts.

► To assign statement permissions

In this procedure, you will assign statement permissions. You will allow the user **kathy** to create views and stored procedures.

1. Start SQL Server Enterprise Manager.
2. In the console tree, expand **Microsoft SQL Servers**, expand **SQL Server Group**, and then expand your server.
3. Expand **Databases**, right-click **Northwind**, click **Properties**, and then click the **Permissions** tab.
4. Select **Create View** and **Create SP** for **kathy**, and then click **OK** to grant the permissions.
5. Log off Windows 2000, and then log on to the **NWTraders** classroom domain by using the information in the following table.

Option	Value
User name	**kathy**
Password	**password**

6. Open SQL Query Analyzer and, if requested, log in to the (local) server with Windows Authentication.

   You have permission to log in to and administer SQL Server because you are logged as **SQLAdmin*x***, which is a member of the Windows 2000 local group, Administrators. All members of this group are automatically mapped to the SQL Server **sysadmin** role.

7. Execute the following Transact-SQL statements to create a view:

```
USE Northwind
GO
CREATE VIEW testview as
  SELECT FirstName, LastName
  FROM Employees
```

   Were you able to create the view?

   Yes, because permissions have been granted to kathy to create views.

   ____

   ____

8. Execute a CREATE TABLE statement:

   ```
   USE Northwind
   CREATE TABLE testtable
     (column1 INT NOT NULL,
      column2 CHAR(10) NOT NULL)
   ```

   Did the statement execute successfully? Why or why not?

   The CREATE TABLE statement failed because kathy does not have statement permissions that allow her to execute it.

   ____

   ____

9. Log off Windows 2000.

Exercise 3
Assigning Object Permissions

In this exercise, you will use SQL Server Enterprise Manager to assign object permissions to SQL Server security accounts. You will assign permissions for the users, groups, and roles that you created in previous exercises.

► To assign object permissions

In this procedure, you will use SQL Server Enterprise Manager to assign object permissions to roles.

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. Start SQL Server Enterprise Manager.
3. In the console tree, expand **Microsoft SQL Servers**, and then expand **SQL Server Group**.
4. Expand your server, expand **Databases**, expand **Northwind**, and then click **Tables**.
5. In the **Details Pane**, right-click **Categories**, point to **All Tasks**, and then click **Manage Permissions**.
6. Use the following table to grant, revoke, and deny permissions on **Northwind** objects for users and roles.

Object	Account	Permission
Categories	public	Grant all
Customers	kathy	Deny all
Customers	cust_mgmt	Grant all
Customers	public	Revoke all
Employees	public	Revoke all
OrderDetails	public	Revoke all
Orders	public	Revoke all
Products	public	Revoke all

7. Log off Windows 2000.

► To test object permissions

In this procedure, you will test the permissions of users and roles.

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

Option	Value
User name	**carl**
Password	**password**

2. Open SQL Query Analyzer and, if requested, log in to the (local) server with Windows Authentication.

   You have permission to log in to and administer SQL Server because you are logged as **SQLAdmin*x***, which is a member of the Windows 2000 local group, Administrators. All members of this group are automatically mapped to the SQL Server **sysadmin** role.

   Remember that **carl** is a member of the **cust_mgmt** role.

3. Execute each of the following Transact-SQL statements to test permissions for **carl**:

```
USE Northwind
SELECT * FROM Employees
SELECT * FROM Customers
SELECT * FROM Categories
SELECT * FROM Products
```

   Which tables can **carl** query? Which tables is he not able to query? Why?

   He can query the Customers and Categories tables. He has permissions to access the Customers table because he is a member of the cust_mgmt role. He can query the Categories table because it has been granted to the public role. He cannot query the Employees or Products table because permissions for these tables have been revoked from the public role, and he does not have any permission through his own account or any roles to which he belongs.

   ____

   ____

4. Log off Windows 2000.

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

Option	Value
User name	**kathy**
Password	**password**

6. Open SQL Query Analyzer and, if requested, log in to the (local) server with Windows Authentication.

   You have permission to log in to and administer SQL Server because you are logged as **SQLAdmin*x***, which is a member of the Windows 2000 local group, Administrators. All members of this group are automatically mapped to the SQL Server **sysadmin** role.

   Remember that **kathy** is a member of the **cust_mgmt** role.

7. Execute the following Transact-SQL statement to test permissions for **kathy**:

   ```
   USE Northwind
   SELECT * FROM Customers
   ```

   Can **kathy** access the table? Why or why not?

   No. Even though she is a member of the cust_mgmt role that has been granted permissions on this table, kathy's own SQL Server user account has been denied access.

   ____

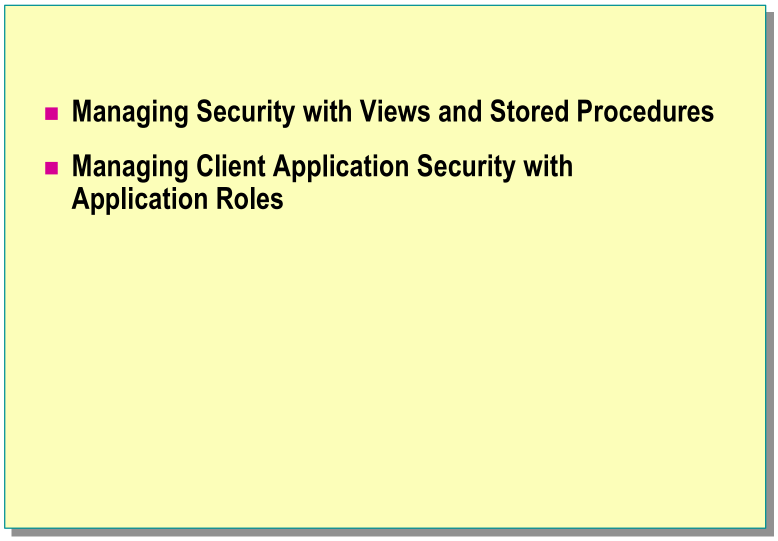
   ____

8. Log off Windows 2000.

◆ Managing Application Security

Topic Objective
To introduce the topic of managing application security.

Lead-in
You can secure access to a database by using login account authentication and permissions. You can also use methods to secure access at the application level, such as views, stored procedures, and application roles.



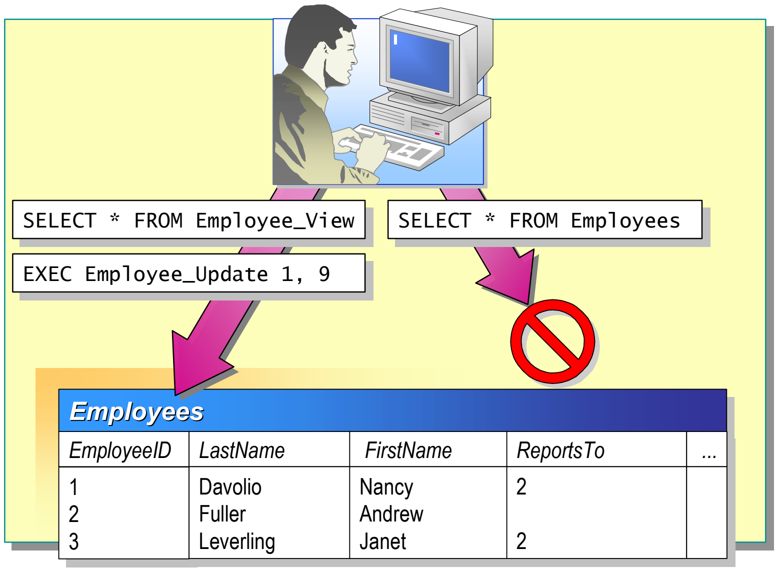
- Managing Security with Views and Stored Procedures
- Managing Client Application Security with Application Roles

You can secure access to a database by using login account authentication and permissions. You can also use methods to secure access at the application level, such as views, stored procedures, and application roles.

Managing Security with Views and Stored Procedures

Topic Objective
To describe how to use views and stored procedures to simplify permissions.

Lead-in
Views and stored procedures provide a secondary method of giving users access to data and the ability to perform activities in a database.



EmployeeID	LastName	FirstName	ReportsTo	...
1	Davolio	Nancy	2	
2	Fuller	Andrew		
3	Leverling	Janet	2	

Delivery Tip
Point out that you can use views to prevent users from knowing that there are certain columns in a table to which they do not have access.

Views and stored procedures provide a secondary method of giving users access to data and the ability to perform activities in a database. They allow you to set up security with SQL Server objects that are created for an application.

Views and stored procedures enable you to manage the permissions on the view or stored procedure only, rather than the permissions on the objects to which they refer. They also shield users from changes to the underlying tables.

To use views and stored procedures to simplify security:

- Grant permissions on a view to users without granting permissions on the underlying tables.

  For example, the **BirthDate** column in a table contains confidential employee information, but the rest of the columns contain information to which users must have access. You can define a view that includes all of the columns in the table with the exception of the **BirthDate** column. As long as the table and the view have the same owner, granting SELECT permissions on the view allows users to view non-confidential columns without permissions on the table itself.

- Grant permissions to execute a stored procedure to users without granting them access to the tables that are modified.

  In an archiving scenario, data that is older than a specified interval is copied into an archive table and then deleted from the primary table. Permissions can be used to prevent users from deleting rows from the primary table directly or from inserting rows into the archive table without deleting them from the primary table. You can create a stored procedure to ensure that both activities are performed together, and then you can grant permissions to users to execute the stored procedure.

You can use SQL Server Enterprise Manager to create views in the design window. Adding selected tables to the diagram pane and selecting columns to display allows you to design views without needing to learn Transact-SQL. Alternatively, you can run the Create View Wizard.

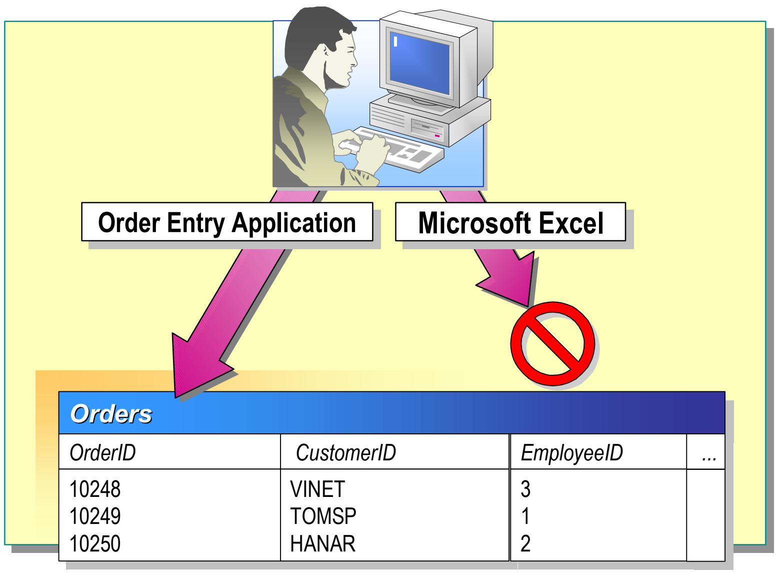
You can create stored procedures in SQL Server Enterprise Manager by using the Create Stored Procedure Wizard. For more complex procedures, you can enter Transact-SQL statements directly into stored procedure definitions.

SQL Query Analyzer provides templates of Transact-SQL scripts to help you create objects in a database, including both views and stored procedures. The scripts contain parameters to help you customize the Transact-SQL scripts, and a **Replace Templates Parameters** dialog box to guide you through the process.

◆ Managing Client Application Security with Application Roles

Topic Objective
To introduce application roles.

Lead-in
Application roles allow you to create a security environment for an application.



OrderID	*CustomerID*	*EmployeeID*	...
10248	VINET	3	
10249	TOMSP	1	
10250	HANAR	2	

Application roles allow you to enforce security for a particular application. They allow you to restrict users to accessing data indirectly, through a specific application only.

For example, you might want order entry clerks to be able to update the **Products**, **Orders**, and **Customer** tables only when they use the order entry application. You do not want the clerks to be able to access the tables from another product, such as Microsoft Excel. In this situation, you could create an application role for the order entry application.

Application roles differ from other roles. The following is a list of the fundamental differences between application roles and other roles:

- Application roles have no members—they are activated for users when the user runs the application.

  This allows application users to have special permissions when they use the application only and avoids the need to grant permissions to users directly.
- Application roles require a password to be activated.

Upon assuming an application role, the user:

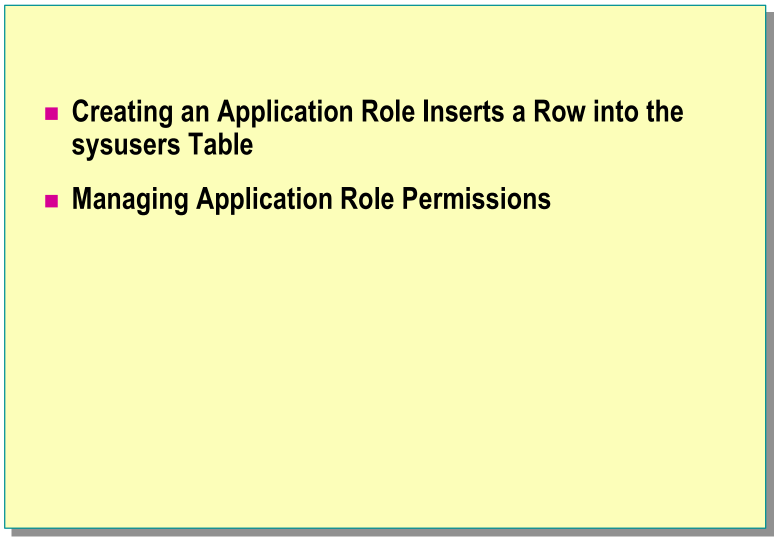
- Loses all existing permissions in the current database for his user account and any roles to which he belongs, except any permissions that apply to the **public** role.
- Inherits all application role permissions in the current database.

The user cannot execute a USE DATABASE statement, but the user can access data in other databases, using fully qualified object names, provided the database has a **guest** account. You must grant permissions to the **guest** account.

Creating Application Roles

Topic Objective
To describe how to create application roles.

Lead-in
You can create application roles and assign permissions to them.



- **Creating an Application Role Inserts a Row into the sysusers Table**
- **Managing Application Role Permissions**

You can create application roles and assign permissions to them.

Creating Application Roles

Use SQL Server Enterprise Manager or the **sp_addapprole** system stored procedure to create a new application role. Only members of the **db_owner**, **db_securityadmin**, and **sysadmin** roles can execute the **sp_addapprole** system stored procedure.

You can create an application role in the same way that you create a user-defined database role, by using SQL Server Enterprise Manager, ensuring that you select **Application role** for the database role type.

Consider the following facts when you create new application roles:

- Creating an application role adds a security account for the new role by inserting a row into the **sysusers** table in the current database.
- The **Password** value is the password that is required to activate the role.

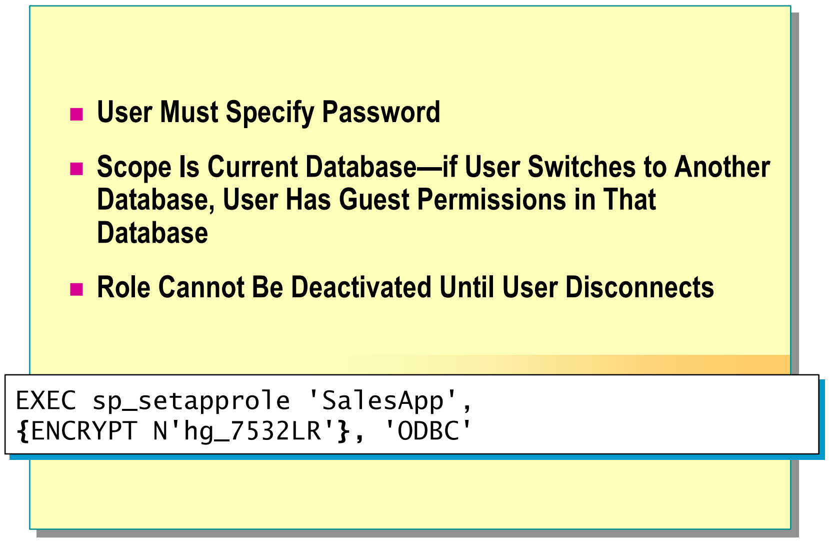
Managing Application Role Permissions

Use SQL Server Enterprise Manager or the GRANT, DENY, and REVOKE statements in Transact-SQL to manage application role permissions.

Activating Application Roles

Topic Objective
To describe how to activate an application role.

Lead-in
Execute the **sp_setapprole** system stored procedure to activate the permissions that are associated with an application role.



- **User Must Specify Password**
- **Scope Is Current Database—if User Switches to Another Database, User Has Guest Permissions in That Database**
- **Role Cannot Be Deactivated Until User Disconnects**

```
EXEC sp_setapprole 'SalesApp',
{ENCRYPT N'hg_7532LR'}, 'ODBC'
```

After a client connects to SQL Server with any login account, the client must execute the **sp_setapprole** system stored procedure to activate the permissions that are associated with an application role. The **sp_setapprole** system stored procedure can only be executed by direct Transact-SQL statements only; it cannot be executed within another stored procedure or from within a user-defined transaction.

Syntax

sp_setapprole [**@rolename** =] '*role*' ,
  [**@password** =] **{Encrypt N** '*password*'} | '*password*'
  [,[**@encrypt** =] '*encrypt_style*']

Consider the following facts when you activate application roles:

- The current application must provide the password, which can be encrypted.
- The scope of an application role is the current database only. If users change to another database, they are allowed to perform activities based on the guest permissions in that database. If no guest user account exists, access is denied.
- After an application role is activated with the **sp_setapprole** system stored procedure, the role cannot be deactivated in the current database until the user disconnects from SQL Server.
- Application roles do not take advantage of Open Database Connectivity (ODBC) connection pooling. Pooling allows an application to use a connection from a pool of connections that do not need to be reestablished for each use. This can improve application performance but relies on the properties of a connection (for example, login name, which determines permissions) remaining unaltered during its lifetime.

It is necessary to supply a password to authenticate the application. Optionally, you can encrypt the password using the ODBC ENCRYPT function. If you encrypt the password, the password must be converted to a Unicode string by preceding the password with the letter N.

You also can use the **sp_dropapprole** system stored procedure to delete an application role from the current database, or the **sp_approlepassword** system stored procedure to change the password for an application role.

Lab C: Managing Application Security

Topic Objective
To introduce the lab.

Lead-in
In this lab, you will learn how to manage application security.

Explain the lab objectives.

Objectives

After completing this lab, you will be able to:

- Create and activate an application role.
- Use views and stored procedures to simplify permissions management.

For More Information

If you require help in 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 that of 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 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
Creating and Activating an Application Role

In this exercise, you will create an application role, assign role permissions, and activate the application role. You will use this role in other exercises later in this lab.

► To define an application role

In this procedure, you will define an application role called **order_entry** in the **Northwind** database by using SQL Server Enterprise Manager. This process is very similar to creating a database role, with the exception that you do not define any members.

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. Start SQL Server Enterprise Manager.

   You have permission to log in to and administer SQL Server because you are logged as **SQLAdmin*x***, which is a member of the Windows 2000 local group, Administrators. All members of this group are automatically mapped to the SQL Server **sysadmin** role.

3. In the console tree, expand **Microsoft SQL Servers**, expand **SQL Server Group**, and then expand your server.
4. Expand **Databases**, and then expand **Northwind**.
5. Right-click **Roles**, and then click **New Database Role**.
6. Create an application role called **order_entry** with a password of **password**.
7. Use SQL Server Enterprise Manager to assign permissions for the **order_entry** application role that you just created. Use the information in the following table.

Table	Permissions
Northwind..Categories	SELECT
Northwind..Customers	SELECT, INSERT, UPDATE
Northwind..Order Details	SELECT, INSERT, UPDATE
Northwind..Orders	SELECT, INSERT, UPDATE
Northwind..Products	SELECT

8. Log off Windows 2000.

► To activate an application role

In this procedure, you will use SQL Query Analyzer to log in as a user and activate the **order_entry** application role.

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

Option	Value
User name	**carl**
Password	**password**

   Remember that **carl** is a member of the **cust_mgmt** role.

2. Open SQL Query Analyzer and, if requested, log in to the (local) server with Windows Authentication.

   You have permission to log in to and administer SQL Server because you are logged as **SQLAdmin*x***, which is a member of the Windows 2000 local group, Administrators. All members of this group are automatically mapped to the SQL Server **sysadmin** role.

3. Execute the **sp_setapprole** system stored procedure to activate the role:

   ```
   USE Northwind
   EXEC sp_setapprole 'order_entry', 'password'
   ```

   Execute the following statement to query the **Products** table:

   ```
   SELECT * FROM Products
   ```

   Were you able to query the table? Why or why not?

   Yes, you can query the table because carl now has only the permissions that are associated with the order_entry role. An application role is exclusive; any other permissions that carl has directly or in other roles in which he is a member will be ignored (except public). For example, carl can do any operation with the Customers table through his membership in the cust_mgmt role, but these permissions are ignored after an application role is activated.

   How long will the **order_entry** role be activated for **carl**?

   The order_entry application role remains active until the session (connection) is closed.

4. Close the query window to end the session. This will also deactivate the **order_entry** role for **carl**.

5. Log off Windows 2000.

Exercise 2
Implementing Permissions by Using Views and Stored Procedures

In this exercise, you will grant permissions for users to use a view and stored procedure to access a table without having permissions to access the table directly.

You will use both SQL Server Enterprise Manager and Transact-SQL statements in this exercise.

► To use a view or stored procedure to implement permissions

In this procedure, you will create a view called **employees_view** and a stored procedure called **employee_proc**. You then will grant permissions on **employees_view** and **employee_proc** to the **public** role and revoke permissions on the **Employees** table from the **public** role.

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. Start SQL Server Enterprise Manager.
3. In the console tree, expand **Microsoft SQL Servers**, expand **SQL Server Group**, and then click your server.
4. On the **Tools** menu, click **Wizards**.
5. Expand **Database**, click **Create View Wizard**, and then click **OK**.
6. Use the information in the following table to create a view.

Option	Value
Database	**Northwind**
Objects	Employees
Columns	**LastName**
	FirstName
	Title
Name	Employees_View

7. Open SQL Query Analyzer and, if requested, log in to the (local) server with Windows Authentication.

   You have permission to log in to and administer SQL Server because you are logged as **SQLAdmin*x***, which is a member of the Windows 2000 local group, Administrators. All members of this group are automatically mapped to the SQL Server **sysadmin** role.

8. Execute the following statement to create a stored procedure that queries the **FirstName**, **LastName**, and **Title** columns of the **Employees** table:

```
USE Northwind
GO
CREATE PROCEDURE employee_proc AS
  SELECT FirstName, LastName, Title
  FROM Employees
```

9. Execute the following statements to allow the **public** role to select from **employees_view** and execute **employee_proc**:

```
USE Northwind
GRANT SELECT ON employees_view TO public
GRANT EXEC ON employee_proc TO public
```

10. Execute the following statement to remove the right of the **public** role to select from the **Employees** table:

```
USE Northwind
REVOKE SELECT ON employees FROM public
```

11. Log off Windows 2000.

► **To test permissions on the view and stored procedure**

In this procedure, you will query data from the **employees_view** view and execute the **employee_proc** stored procedure. You will then attempt to query the **Employees** table directly.

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

Option	Value
User name	**carl**
Password	**password**

2. Open SQL Query Analyzer and, if requested, log in to the (local) server with Windows Authentication.

   You have permission to log in to and administer SQL Server because you are logged as **SQLAdmin*x***, which is a member of the Windows 2000 local group, Administrators. All members of this group are automatically mapped to the SQL Server **sysadmin** role.

3. Execute the following statement to query the **employees_view** view:

```
SELECT * FROM employees_view
```

   Were you able to query the view? Why or why not?

   Yes, you can query the view even though carl does not have permissions on the underlying table, because carl has permissions to select from the view through membership in the public role.

   ____

   ____

4. Execute the **employee_proc** stored procedure:

   ```
   EXEC employee_proc
   ```

   Were you able to execute the stored procedure? Why or why not?

   Yes, you can execute the stored procedure even though carl does not have permissions on the underlying table, because carl has permissions to execute the stored procedure via membership of the public role.

5. Execute the following statement to query the **Employees** table:

   ```
   SELECT * FROM Employees
   ```

   Were you able to query the table? Why or why not?

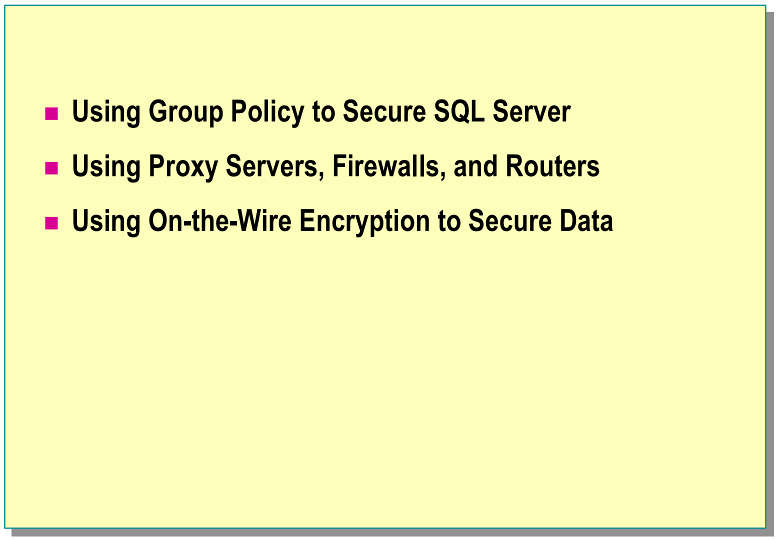
   No. carl only has permissions that allow him to use the employees_view view and the employee_proc stored procedure. SELECT privileges have been specifically revoked from the public role.

6. Log off Windows 2000.

◆ Managing SQL Server Security in the Enterprise

Topic Objective
To introduce the extra security considerations needed when using SQL Server in the enterprise.

Lead-in
In this section, we will discuss some of the extra security considerations necessary when working in the enterprise.



- **Using Group Policy to Secure SQL Server**
- **Using Proxy Servers, Firewalls, and Routers**
- **Using On-the-Wire Encryption to Secure Data**

When using SQL Server in the enterprise environment, you must consider some additional security issues. You can use Group Policy to provide security configurations for multiple users or computers. Proxy servers, firewalls, and routers can secure your internal network from unauthorized external access, and you can use on-the-wire encryption facilities within Windows 2000 to ensure that data packets cannot be read, should they be accessed.

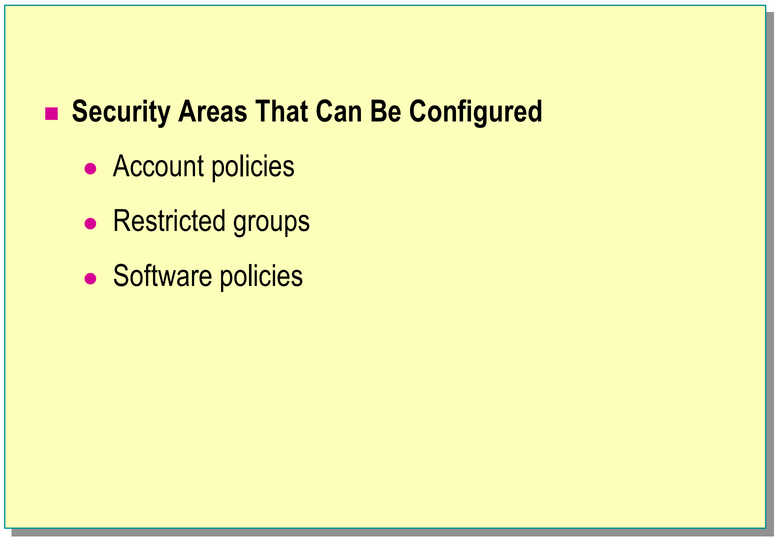
Note This section is a high level overview of subjects related to security in an enterprise environment.

For more information about the topics in this section, see *Blueprint for Building Web Sites Using the Microsoft Windows DNA Platform* on the Student Materials compact disc.

Using Group Policy to Secure SQL Server

Topic Objective
To introduce Group Policy in Windows 2000.

Lead-in
In Windows 2000, you can use Group Policy to define…



- **Security Areas That Can Be Configured**
  - Account policies
  - Restricted groups
  - Software policies

In Windows 2000, you can use Group Policy to define user and computer configurations for groups of users and computers. You can use Group Policy for configuring many options, including registry-based policies, security options, and software installation.

The security areas that you can configure for computers include:

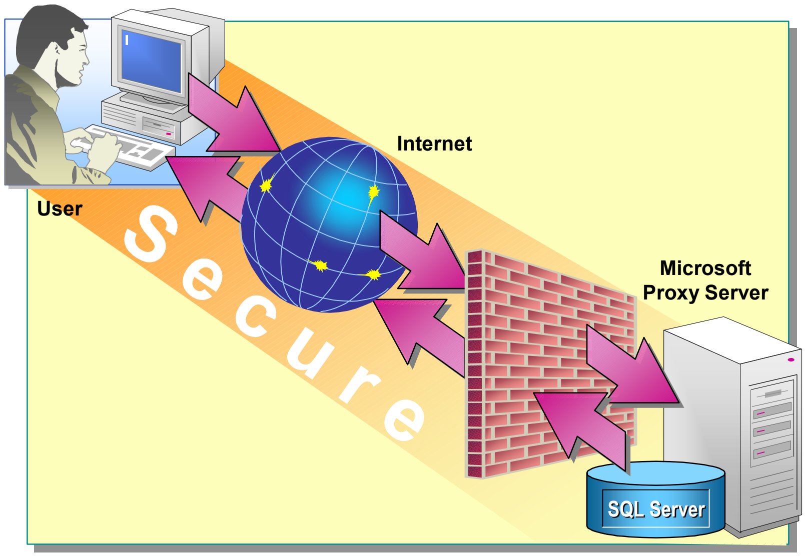
- *Account policies*. Account policies are computer security settings for password policy, lockout policy, and Kerberos policy in Windows 2000 domains.
- *Restricted groups*. Restricted groups allow you to control who belongs to a specific group, as well as the groups to which a restricted group should belong. Restricted groups allow administrators to enforce security policies regarding sensitive groups, such as Enterprise Administrators or Payroll.

  For example, you may decide that only Joe and Mary should be members of the Enterprise Administrators group. You can use restricted groups to enforce that policy. If you add a third user to the group (for example, to accomplish some task in an emergency situation), the next time that the policy is enforced, that third user is automatically removed from the Enterprise Administrators group.
- *Software policies*. You can use Group Policy to define default settings that will be automatically applied to users and computers. These settings can determine security options and control what software is available to particular groups of users.

Using Proxy Servers, Firewalls, and Routers

Topic Objective
To discuss the purpose of proxy servers, firewalls, and routers.

Lead-in
Connecting your SQL Server to the Internet can create additional security concerns.



Connecting your SQL Server to the Internet can create additional security concerns. You need to ensure that only authorized users can access the database server, and that they can access only the resources required for the business process.

Connections to SQL Server Through Proxy Server

You can allow connections to SQL Server through Microsoft Proxy Server, a stand-alone application that provides secure access to and from the Internet. This allows you to prevent unauthorized users from connecting to your private network. It keeps your sensitive data secure by controlling all permissions and access to the listening port.

Firewall Security

Firewalls are one of the best ways to protect your system from attack by users on the Internet. A *firewall* restricts both inbound and outbound access, either by granting permissions to Windows users or by disallowing data to be passed through specified TCP/IP ports. They also can analyze all traffic between your network and the Internet.

You can use the firewall security features of Proxy Server to control the flow of information to and from Proxy Server. Proxy Server supports both inbound and outbound packet filtering. It dynamically determines which packets can be passed through to the internal network's circuit and application layer proxy services. Individual packet filters are configured to prevent any packets except for the ones specified from being passed through Proxy Server. This allows ports to be opened automatically, only as needed, and then closed when the communication ends. This practice minimizes the number of exposed ports in either direction and provides high-level security for your network.

Web sites are usually accessed on port 80, and you should enable this port so that Internet users can access your Web site. If the site uses Active Server Pages (ASP) or some other server-side scripting mechanism to access data from SQL Server and return it to the client as Hypertext Markup Language (HTML), then you need not enable any other port.

Many Web-based applications use two firewalls. The first firewall separates the Web server from the Internet and allows access only through port 80. The Web server resides in a semi-secure area called a *perimeter network*, and accesses the SQL Server database through a second firewall between the perimeter network and the corporate network. The official Internet Assigned Numbers Authority (IANA) socket number for SQL Server is 1433 (although you can configure this by setting a registry key). When configuring access to SQL Server by using TCP/IP, it is necessary to let data through this port. A named instance of SQL Server will be dynamically assigned a port when the instance is started, but if you want to use proxy services with a named instance you should override this and assign a fixed port to the named instance by using SQL Server Network Utility. If communicating through COM+, port 135 must be enabled along with a range of allowable values that the COM+ protocol uses to negotiate connectivity with the client.

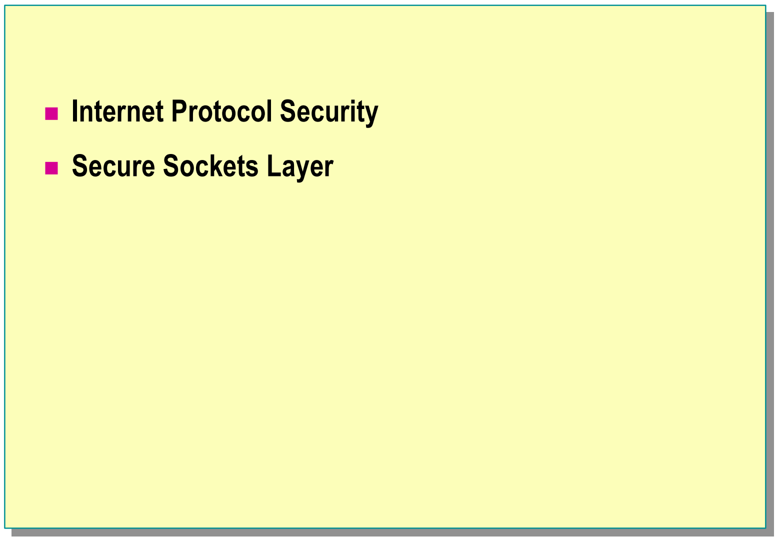
Routers

Routing features built into Windows 2000 servers allow the servers to act as routers. A *router* (or gateway) is a device that forwards Internet Protocol (IP) packets from one network to another. You can use a router in conjunction with a firewall to configure data traveling in and out of your network.

Using On-the-Wire Encryption to Secure Data

Topic Objective
To introduce the topic of on-the-wire encryption.

Lead-in
When transmitting data packets across a network or the Internet you must ensure that packets cannot be read if they are accessed.



- **Internet Protocol Security**
- **Secure Sockets Layer**

When transmitting data packets across a network or the Internet you must ensure that packets cannot be read if they are accessed.

Internet Protocol Security

Windows 2000 incorporates Internet Protocol Security (IPSec) for data protection of network traffic. It is a set of Internet standards that uses cryptographic security services to provide:

- *Confidentiality*. IPSec traffic is encrypted. Captured IPSec traffic is unintelligible without knowledge of the encryption key.
- *Authentication*. IPSec authenticates the sender of IP data packets on the basis of Kerberos authentication, digital certificates, or a shared secret key.
- *Data integrity*. IPSec traffic contains a cryptographic checksum that incorporates the encryption key. The receiver can verify that the packet was not modified in transit.

You can configure IPSec security policy for each domain or for each local computer by defining a set of rules and filters that are to be applied to regulate secure communication with specific IPSec clients.

Secure Sockets Layer

SQL Server can use the Secure Sockets Layer (SSL) to encrypt all data transmitted between a client and a server.

SSL encryption works only with instances of SQL Server that have been assigned a certificate from a public certification authority (CA). Server certificates allow the client to positively identify the server before sharing sensitive information. The client computer must also have a root CA certificate from the same authority. Client certificates contain personal information about the clients requesting access to your server.

Information within the certificates is used to encrypt data. This ensures that only the server and client are able to decrypt the information.

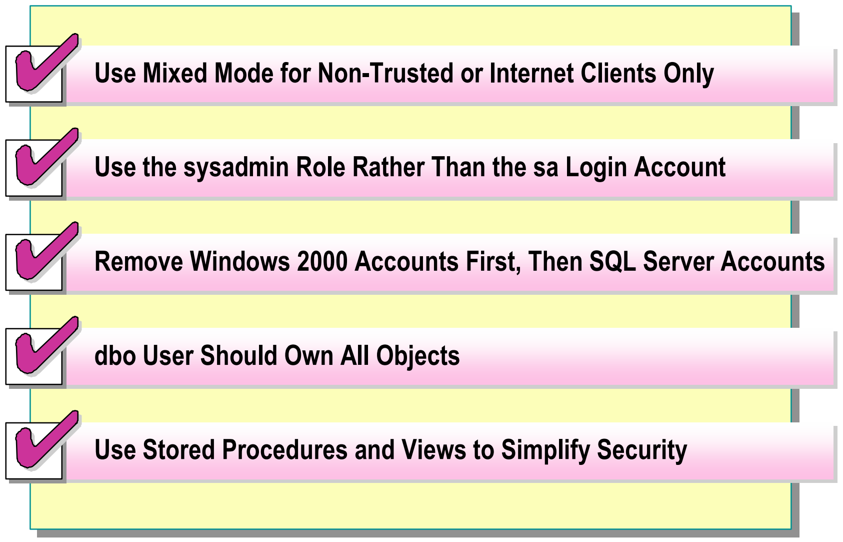
You can enable SSL encryption for all of the enabled server protocols by using SQL Server Network Utility. The encryption is turned on or off for the entire enabled server protocols, and you cannot specify encryption for a specific protocol.

Note For compatibility with earlier versions of SQL Server, the Multiprotocol Net-Library continues to support its own encryption. This encryption is specified independently of the SSL encryption and does not require the use of certificates. This is not supported by named instances.

Recommended Practices

Topic Objective
To list the recommended practices for implementing security.

Lead-in
Consider the following recommendations when you implement security.



- Use Mixed Mode for Non-Trusted or Internet Clients Only
- Use the sysadmin Role Rather Than the sa Login Account
- Remove Windows 2000 Accounts First, Then SQL Server Accounts
- dbo User Should Own All Objects
- Use Stored Procedures and Views to Simplify Security

Consider the following recommendations when you implement security:

- Use Windows Authentication Mode if your network environment supports only trusted client connections. Use Mixed Mode to connect non-trusted or Internet clients only.
- Use the System Administrators (**sysadmin**) role rather than the **sa** login account.
- When you remove Windows 2000 users or groups, first remove them from Windows 2000 in order to disallow network access. Then, remove them from SQL Server.
- All objects that are intended for general use should be created with the **dbo** user so that they are easily accessible.
- Using stored procedures and views to implement security can be more efficient than implementing column-level security, from both administrative and performance perspectives.

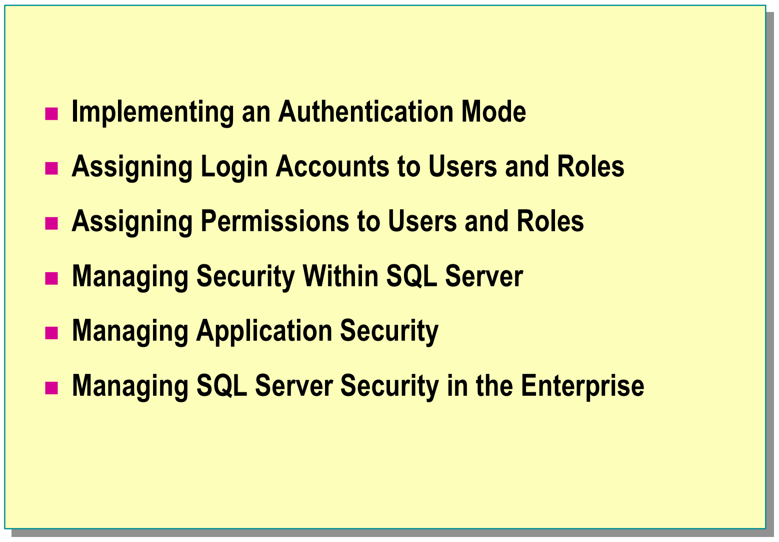
Additional information on the following topics is available in SQL Server Books Online.

Topic	Search on
Security-related system stored procedures	“security procedures”
Planning security	“planning security”
Advanced security	“advanced security”

Review

Topic Objective
To reinforce module objectives by reviewing key points.

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



- **Implementing an Authentication Mode**
- **Assigning Login Accounts to Users and Roles**
- **Assigning Permissions to Users and Roles**
- **Managing Security Within SQL Server**
- **Managing Application Security**
- **Managing SQL Server Security in the Enterprise**

1. What type of authentication mode would you implement in an environment that contains users who connect from both UNIX and Windows 2000? Why?

   You should use Mixed Mode because it supports both Windows Authentication and SQL Server Authentication.

2. When should you assign permissions to a SQL Server user account directly?

   You should assign permissions to a SQL Server user account directly when that user account maps to a Windows 2000 group that requires common permissions.

3. If permissions to update a table are granted to a user but the permissions were denied to a role in which the user has membership, does the user account retain permissions to update the table?

   No. Denial of permissions supersedes granting of permissions.