

Security: Protecting Against Peeping Toms!

In This Chapter

- Securing your databases
- Adding new logins
- Taking on new roles
- Taking a different slant with views

Not that you shouldn't trust your fellow man, but you always need to consider security as an issue. SQL Server 2000 has lots of features that help you to secure your data. If you don't secure your data, anyone can get access to it. Do you want someone to know the salaries of every employee in the company? If not, read on.

Security Considerations

There are three basic areas of security that you need to consider: who has the ability to log into SQL Server; who has access to your databases; and security on objects in those databases.

Server access

Before any user can access a database or objects within that database, a user or group must be able to log into SQL Server. After this user or group has a login into SQL Server, you then determine which databases the login has access to.

When you create a login (which is shown later in this chapter), you have the choice of authentication modes. You can use Windows authentication or SQL Server authentication. I describe each of these authentication modes in the next two sections. This issue can become quite confusing because you can specify the authentication mode in two places. You can specify it at the server level and at the login level.

At the server level, SQL Server 2000 allows for two authentication modes: SQL Server and Windows, also known as **mixed mode**, as well as Windows only. The bottom line is that if mixed-mode authentication is chosen at the server level, then Windows NT or Windows 2000 can be used to authenticate the login, or SQL Server can be used to authenticate the login. However, if Windows only is chosen as the authentication mode, you cannot have SQL Server authenticate a login, no matter what!

Although both of these topics are beyond the scope of this book, I want you to be aware of two security concepts available in SQL Server 2000. The first is Kerberos. Kerberos is a much more secure and robust security model than the Windows NT model, known as NTLM, which is also known as challenge/response.

The second concept is security delegation. If all servers are running SQL Server 2000 with Kerberos support enabled, your Windows 2000 Active Directory can be configured to use delegation, which specifies that multiple servers used in queries use the same security settings as the server that you are logged into.

Server-level authentication modes can be specified when you install SQL Server 2000 or afterwards by right-clicking on the name of your server in the Enterprise Manager and choosing the Properties menu. Then, clicking the Security tab brings up the dialog box shown in Figure BC2-1, allowing you to configure the server authentication mode.

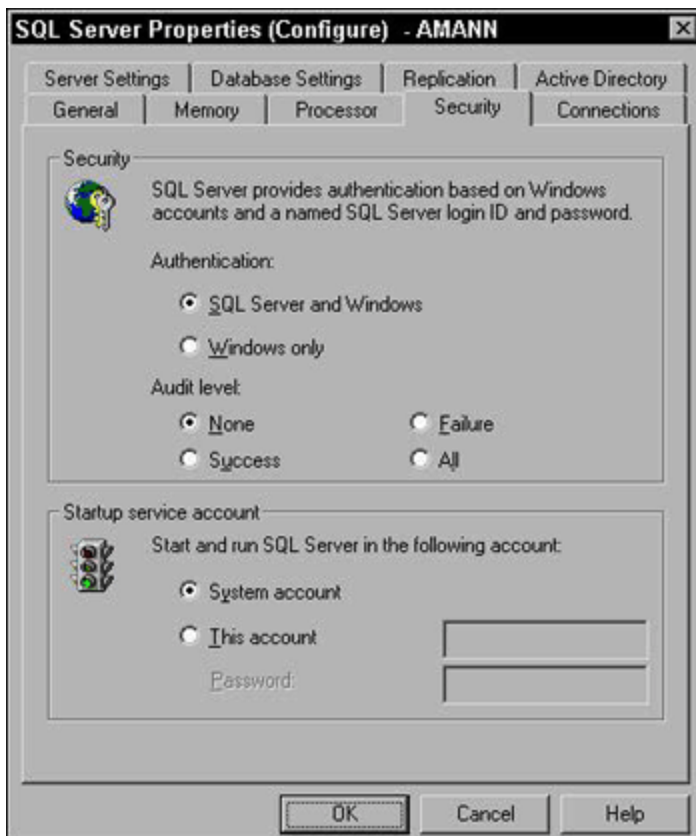


Figure BC2-1: Selecting the server authentication mode with the Enterprise Manager.

Windows authentication -- login level

Windows authentication specifies that the login information will be authenticated by a Windows NT or Windows 2000 domain controller. SQL Server 2000 uses a trusted connection because it **trusts** that Windows NT or Windows 2000 authenticated the login. Using Windows authentication does not have to be done with a specific user name only. You can use any valid user name or group.

When the user logs into the operating system, Windows NT or Windows 2000 authenticates the login and determines what groups the user is a member of. If you specify that a specific Windows NT group can log into SQL Server, any user who is a member of the group and has successfully logged into the operating system will have access to the specified server, as well as any databases within that server that

have been deemed to be accessible. See the "Database access" section for more information on this.

The Windows authentication mode is not available if the server is running on a Windows 95 or Windows 98 computer.

Windows Authentication is always used if the server authentication mode is set to Windows only.

SQL server authentication -- login level

With SQL Server authentication, Windows NT or Windows 2000 accounts are not involved at all. SQL Server 2000 stores all the login information and passwords. The drawback to using SQL Server authentication is that even though a user has logged into the operating system, he/she must also log into SQL Server. I recommend not using SQL Server authentication if you can avoid it because accounts need to be managed in two places (in the Windows NT or Windows 2000 domain for logging into the network and SQL Server for logging into SQL Server 2000).

Even though you can create logins with SQL Server Authentication, they will not be used unless you choose mixed-mode authentication at the server level.

Database access

Without access to a database, a user or group cannot use a database or any objects within that database. Therefore, after a user is authenticated to gain access to SQL Server, you must specify which database(s) will be available for the login or group. You can do this at the same time the login is added, as you see later in this chapter.

Object permissions

After a user has been given a login into SQL Server and has been given access to a database, you must also specify which objects the login or group has permissions to access. These objects can be tables, views, or stored procedures. There is not just one permission type for an object, there are several (depending on the type of object you're dealing with). Each of the permission types listed in Table BC2-1 can also specify a permission level. I discuss permission levels in the next section. SQL Server 2000 gives you the ability to further refine the permissions on a table or a view, but allows you to specify permissions on each column within the table.

If you don't explicitly specify permissions on columns within a table or a view, all columns will have the same permissions as they are defined on the table or view that contains the columns.

Table BC2-1: Permission Types Available for Database Objects

Permission Type	Object Availability	Description
SELECT	Tables and Views, as well as columns within those Tables and Views	Specifies the permission level for the ability to select data from a table or view
INSERT	Tables and Views	Specifies the permission level for the ability to insert data into a table or view
UPDATE	Tables and Views, as well as columns within those	Specifies the permission level for the ability to update existing data in a table or view
DELETE	Tables and Views	Specifies the permission level for the ability to remove or delete data from a table or view
EXECUTE	Stored Procedures	Specifies the permission level for the ability to run, or execute a stored procedure
DRI	Tables and Views	Specifies the permission level for the ability to specify Declarative Referential Integrity, or DRI. DRI is the explicit declaration of how tables relate to each other by foreign keys and references

It is much easier to administer permissions for a group of users by creating roles within SQL Server 2000. You can then assign permissions on objects or groups of objects to a role. Then, all you have to do is add Windows NT or Windows 2000 users or groups to those roles because permissions have already been set up.

If you can follow this example, then you understand SQL Server 2000 security. Suppose that you have two Windows 2000 users, amann and jsmith. Each of these users belongs to the Windows 2000 group IT Department. Using Windows authentication, you add a new login to SQL Server and specify the login ID as the IT Department Windows 2000 group. You can then assign object permissions for the IT Department group. Now, anytime you add or remove users in the Windows 2000 domain to/from the IT Department group, they are automatically given the appropriate permissions to access the database and objects within that database. It sounds complex, but it really isn't.

For each of the permission types listed in Table BC2-1, you can specify a permission level. Table BC2-2 shows you those permission levels.

Table BC2-2: Permission Levels Available for Permission Types

Permission Level	Description
GRANT	Explicitly grants the permission type on an object
REVOKE	Explicitly revokes a previously explicitly granted or denied permission type on an object
DENY	Explicitly denies the permission type on an object

Permissions are said to be cumulative. This means that if you belong to more than one group for which permissions are defined, you have the highest level of permissions available in all those groups, with the exception of DENY. If you belong to any group that has explicitly been denied access to an object, then you will be denied access, regardless of other permissions in other groups.

What does all this mean? The best way to show you is to provide an example. I have a Windows 2000 domain login, named amann. I have been given SQL Server access using Windows authentication. I have also been given access to the SALES database. I belong to the public role. The public role has been granted (using the GRANT permission level) EXECUTE permissions on the usp_CalculateTotal stored procedure in the SALES database.

Now, I can log into my computer and have the Windows 2000 domain controller validate my login. After I log into the operating system, SQL Server 2000 uses the trusted connection (from Windows NT authentication) to allow me to access the SALES database, where the usp_CalculateTotal stored procedure exists. Because I have been granted EXECUTE permissions on this object, am I allowed to execute it? Well, that depends. If the usp_CalculateTotal stored procedure needs to access tables, I need to be granted SELECT, INSERT, UPDATE, or DELETE permissions (as appropriate) on those tables as well. If SQL Server 2000 allowed me to access the underlying tables without permissions to do so, there would be a hole in the security model.

Adding a New Login

As discussed previously, anyone connecting to SQL Server must have a login ID and password. This ID and password combination is referred to as a **login**. The database administrator (DBA) is usually the person who sets up users in SQL Server. Regardless of who adds new logins, the individual must be a member of either the sysadmin or securityadmin fixed server roles. These roles are described later. The built-in sa login account is, by default, a member of the sysadmin role.

Like most things in SQL Server, setting up logins is quite simple. You can create a new login with the Create Login Wizard or by using the Enterprise Manager.

Creating a login with the Create Login Wizard

The Create Login Wizard guides you through a series of steps in creating logins, prompting you for input along the way (see Appendix A for a flowchart of the steps). To create a login by using the Create Login Wizard, follow these steps:

- 1. Choose Start-Programs-Microsoft SQL Server-Enterprise Manager to start the SQL Server Enterprise Manager (see Figure BC2-2).**

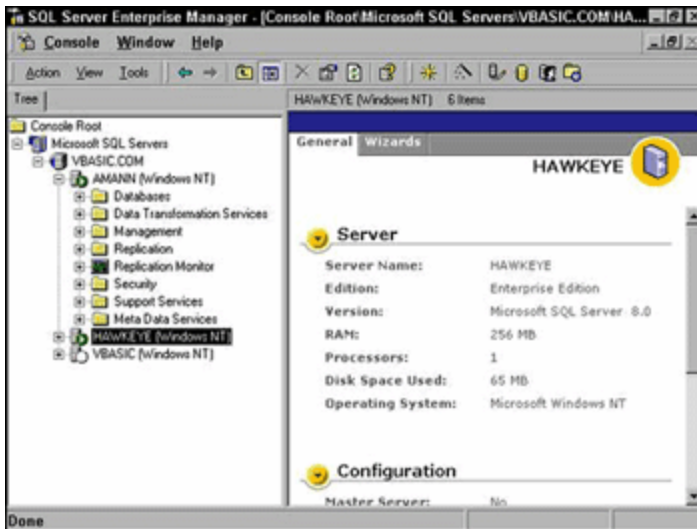


Figure BC2-2: The SQL Server 2000 Enterprise Manager.

2. Expand the tree so that you see the name of your server.

For more information about expanding the tree, see the Introduction of this book.

3. Bring up the Select Wizard dialog box by choosing Tools-Wizards.

The Select Wizard dialog box appears (see Figure BC2-3).

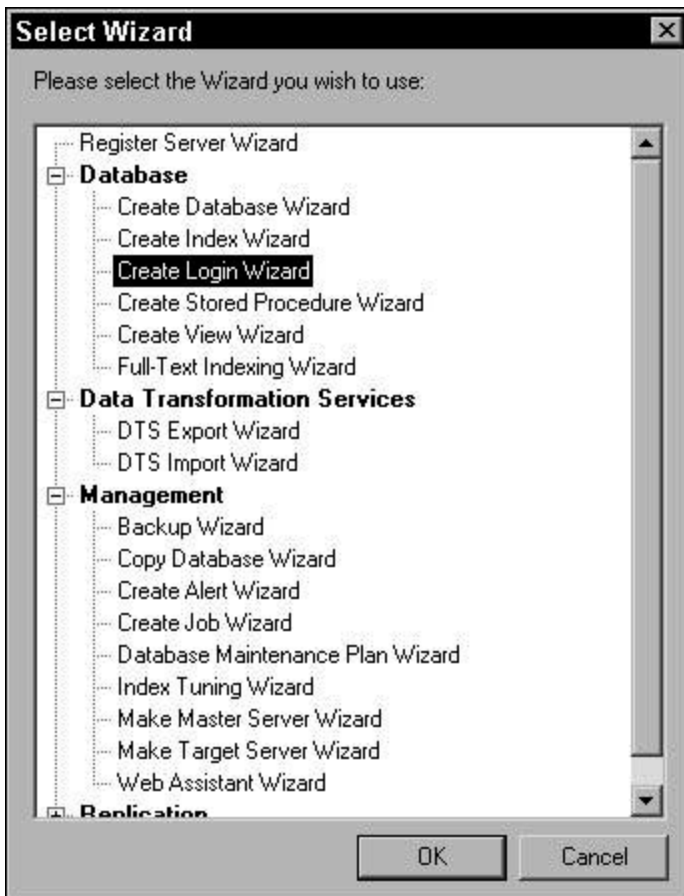


Figure BC2-3: The Select Wizard dialog box.

4. Select the Create Login Wizard item (under the Database category) with the left mouse button and click OK.

The Create Login Wizard prompts you to create your login by asking you a series of questions. These questions are presented in steps. The first step is an introduction dialog box letting you know what the wizard will do for you. Click the Next button when you are ready to begin.

5. For the Select Authentication Mode step that appears, specify the desired mode and click Next (see Figure BC2-4).



Figure BC2-4: Selecting the authentication mode.

Select the desired authentication mode: Windows authentication or SQL Server authentication. See the previous section "Server Access" to help determine which option should be chosen. Windows authentication is chosen by default.

6. In the next step (shown in Figure BC2-5), specify the login information and then click Next.



Figure BC2-5: Selecting login information.

Depending on whether you chose Windows authentication or SQL Server authentication, you are prompted for different information. For Windows authentication, you specify the Windows NT or Windows 2000 domain, followed by a backslash (\) and then the user name in the domain. The user must already exist in the domain before the login can be created. You also specify if the login is granted or denied access, with GRANT being the default option.

SQL Server authentication requires you to specify a login ID as well as a password.

7. For the Grant Access To Security Roles step that appears, click all the desired fixed server roles presented for the new login; then click Next.

You see a list of all possible fixed server roles (see Figure BC2-6). Select the desired fixed server roles for the new login. See the discussion "Selecting server roles," later in this chapter.

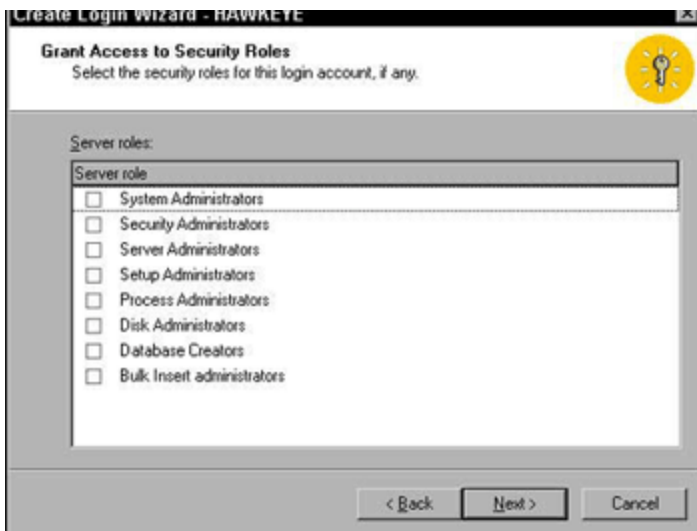


Figure BC2-6: Granting access to fixed server security roles.

8. For the Grant Access To Databases step that appears (see Figure BC2-7), click all the desired databases that the login is to be permitted to use and click Next.



Figure BC2-7: Granting access to databases.

9. Review your choices in the Completing the Create Login Wizard dialog box. If you like what you see and it's exactly what you want, click Finish.

To complete the Create Login Wizard, review the parameters that you have chosen for the new login. If you want to change any information, click the Back button.

Creating a login with the SQL Server Enterprise Manager

The Enterprise Manager enables you to use a graphical program to set up logins easily, just like using the Create Login Wizard, instead of going through the process of issuing SQL statements. If you're a SQL Server "old-timer," you may opt to use SQL statements, which are still available (but not the subject of this chapter).

Here's how you can add a new login with Enterprise Manager:

1. Start the SQL Server Enterprise Manager by choosing Start-Programs-Microsoft SQL Server-Enterprise Manager (refer back to Figure BC2-2).

2. Expand the tree so that you see the Logins folder under the Security folder; then select the Logins folder by clicking it.

For more information about expanding the tree, see the Introduction of this book.

3. Choose Action-New Login.

Alternatively, you can right-click the Logins folder, or click anywhere in the right pane and then click New Login.

Any of these methods brings up a dialog box consisting of three tabs (General, Server Roles, and Database Access), to enable you to type and select the parameters and data relating to the new login (see Figure BC2-8). The General tab appears by default.

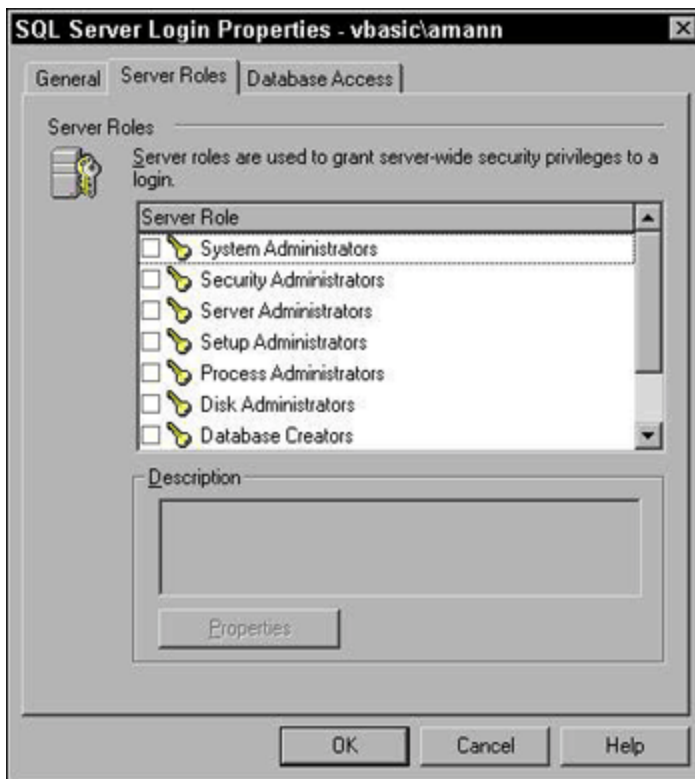


Figure BC2-8: The General tab of the SQL Server Login Properties dialog box.

4. After you type or select the appropriate data, as discussed in the next three sections, click OK to save changes and close the dialog box.

Entering general data for the login

In the General tab, you enter data that describes the login as a whole. General data includes information like the new user's name and security information. Fill in the following fields to enter data in the General tab:

Name: Give the new user's login a name, which is usually referred to as the login ID. It's a good idea to use some type of convention when assigning names, such as the combination of first initial and last name -- for example, tmann. Also consider how you want to use capitalization. All lowercase or all

uppercase letters are frequently used. If you are selecting an existing name from a Windows NT or Windows 2000 domain, you can select the ellipses (·) button and choose the existing login. This login does not have to be an individual. It can be a group.

Windows Authentication: Choose this option if you want a Windows NT or Windows 2000 Server acting as a domain controller to verify the user name and password. This is useful in an Enterprise situation when all authentication occurs from one server. See the discussion earlier in this chapter. Choosing this option enables the Domain drop-down list and Security Access options.

Choose a domain name from the drop-down list or type the domain name that you want to authenticate the user's login ID. Then click the option button corresponding to the desired Security Access. Choosing the Grant Access option grants access to SQL Server for this user, if authenticated. Clicking the Deny Access option denies access to SQL Server for this user, if authenticated.

SQL Server Authentication: Click this option if you want a SQL Server to verify the user name and password. Again, see the section "Server Access," earlier in this chapter. This option is useful when you want a system to be stand-alone. Choosing this option enables the Password text box, where you can type the desired password.

Database: Choose from the drop-down list of available databases to indicate your preferred default database. A user has access to any database that he or she has permissions for. However, if the user never specifies a database, the default database chosen here is used.

Language: Choose from the drop-down list of available languages to indicate the desired default language. A user has access to any language within SQL Server. However, if you never specify a language, the default language chosen here is used. The default language is chosen when you install SQL Server 2000.

Selecting server roles

You can select server roles by using the Server Roles tab, as shown in Figure BC2-9.



Figure BC2-9: Selecting fixed server roles in the SQL Server Login Properties dialog box.

A **server role** is a container for which specific permissions are assigned. These server roles are permitted to perform specific actions within the server as a whole; not in any specific database. For example, a user can have limited permissions in a database -- but have the ability to add new logins. Although you probably won't encounter this situation often, just know that it's possible.

The server roles are presented in a list. You can scroll down to see all the available roles that you can assign to a login. You can assign all, none, or anywhere in between. To select a role, click the check box to the left of the role listed. A role is selected if you see a check mark in the check box.

You cannot add a new server role. You can add only database roles. That is why server roles are sometimes referred to as **fixed server roles** (because they are fixed and cannot change). For more information about adding new roles, see the "Adding New Roles" section, later in this chapter.

The following seven roles are available when you install SQL Server 2000:

- **System Administrators** enables a login to act as a system administrator.
- **Security Administrators** enables a login to manage user logins.
- **Server Administrators** enables a login to manage SQL Server 2000 settings.
- **Setup Administrators** enables a login to install replication and manage extended procedures (see Chapter 8 for a discussion of stored procedures).
- **Process Administrators** enables a login to manage SQL Server processes.
- **Disk Administrators** enables a login to manage disk files.
- **Database Creators** enables a login to manage databases.

By default, no role is selected. You can choose to leave each check box unchecked.

If you want to view any of the actual commands or permissions available to a role, click the Properties button to bring up a dialog box. Then click the Permissions tab to display a read-only list of operations that the selected server role can perform. You can also add members (logins) to the server role by clicking the General tab, followed by the Add button.

Selecting database access

You can control a login's database access by using the Database Access tab (see Figure BC2-10). The tab is used to assign the databases and the privileges that a login has access to within those databases. Click the tab to bring it forward and then fill in the following fields to indicate general database access preferences:

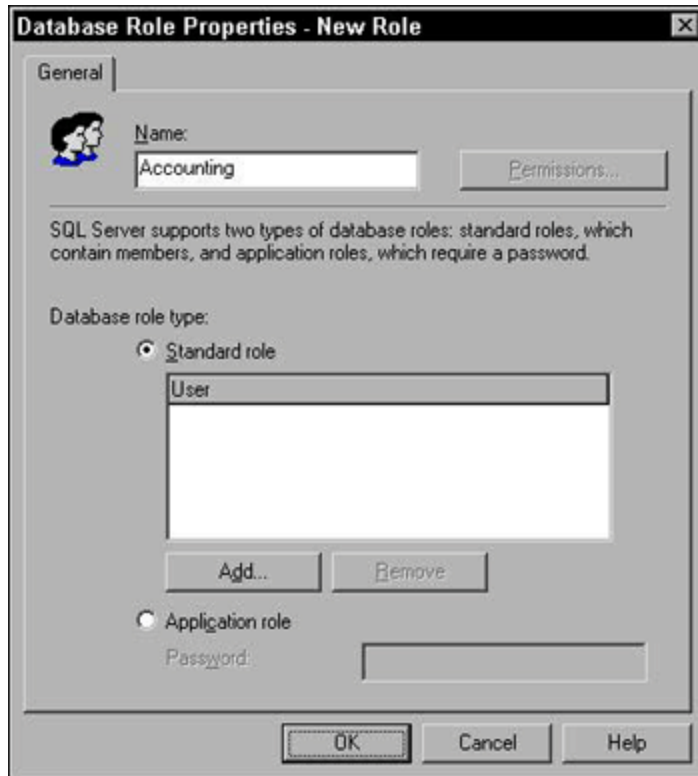


Figure BC2-10: The Database Access tab of the SQL Server Login Properties dialog box.

Specify Which Databases Can Be Accessed by This Login:

All databases for the currently selected SQL Server are presented in a list box. Give a login access to a specific database by scrolling down the list of databases and then clicking the Permit column check box that corresponds to the chosen database. The database is selected when you see a check mark in the Permit column check box.

Database Roles for x: X is not actually presented on the dialog box. X indicates the name of the database, as chosen in the Login grid at the top of Figure BC2-10. Specify which databases can be accessed by this Login section at the top of Figure BC2-10. When a database is selected with a check mark in the Permit column, possible database roles are listed. You can then choose from the list of available database roles. Note that these roles

are not the same as fixed server roles. Database roles can be given to a login within a specific database; fixed server roles are system wide. You can add your own roles (discussed in the next section), but the default database roles are the following:

- **public** enables the login to have the same access as any other public user. All logins are members of the public role by default.
- **db_owner** enables the login to act as the owner of the database and has all permissions.
- **db_accessadmin** enables the login to act as the database administrator, allowing the login to add or remove Login IDs.
- **db_securityadmin** enables the login to act as the security administrator. This allows a login to manage all permissions, object ownerships, roles, and role memberships.
- **db_ddladmin** enables the login to issue Data Definition Language (DDL) statements. I cover DDL statements in Chapter 9. However, the login cannot issue GRANT, REVOKE, or DENY statements.
- **db_backupoperator** enables the login to perform database backups, also known as dumps. This allows a login to issue DBCC, CHECKPOINT, and BACKUP statements.
- **db_datareader** enables the login to read data by granting SELECT permissions on any object.
- **db_datawriter** enables the login to write data. The login can grant INSERT, UPDATE, and DELETE permissions on any object.
- **db_denydatareader** disallows the login from reading data by denying SELECT permissions
- **db_denydatawriter** disallows the login from writing data by denying INSERT, UPDATE, and DELETE permissions.

Adding New Roles

A role is a way to group together permissions for multiple logins, which simplifies administering SQL Server because you can assign permissions to a role and then assign logins to the role. Every time you add a new login, you just assign the login to the role. You don't have to specify permissions. See earlier discussion and examples of roles in this chapter.

The easiest way to set up roles is through the Enterprise Manager because you can use a graphical program rather than issue SQL statements. SQL Server veterans may prefer to use SQL statements -- the option is still available, but I don't address the subject in this section.

To add a new role, follow these steps:

1. To start the SQL Server Enterprise Manager, choose Start-Programs-Microsoft SQL Server-Enterprise Manager.
2. Find the desired database on the expanded tree.

For more information about expanding the tree, see the Introduction of this book.

3. Drill-down the tree within your database until you see the Roles folder.
4. Choose Action-New Database Role to create a new role.

Alternatively, you can right-click the Roles folder or anywhere in the right pane and choose the New Database Role menu option.

Any method you choose to create a new role brings up a dialog box consisting of only one tab. This tab enables you to type or select the parameters and data relating to the new role, as shown in Figure BC2-11.

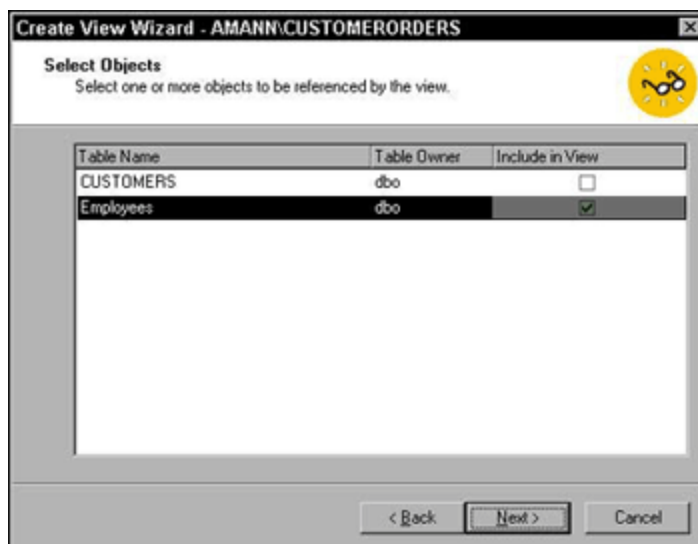


Figure BC2-11: Adding a new database role.

To enter data in the General tab, fill in the following fields:

Name: You may want to relate the new role's name to the logins and permissions that you expect to grant. Say, for example, the role will affect all people in the accounting department. Give the role an easy-to-remember name -- Accounting!

Standard Role: Click this option if you want to assign specific logins to this role. When the option is chosen, you have the opportunity to assign logins by clicking the Add button. This option is chosen by default.

Application Role: Rather than assign specific logins to this role, you can click this option to have logins enter a password. When you choose this option, the Password text box becomes enabled. Type your password into this text box. An application role is useful when you don't want specific users have to log into a database but you do want an application to have access to the database. The application

role prevents unauthorized access to a database by other applications. For example, you can write an application that accesses SQL Server, but you want to avoid Microsoft Access (another application) from getting to your data. Therefore, you have an application role assigned whereby the application logs in using the role name and password.

5. Click the OK button to save changes and close the dialog box.

Creating a New View

A **view** is an important concept in SQL Server. You can use a view to show a login, or a group of logins, only the data that you want him or her to see. For example, you may want a Human Resources manager to have access to data in every column of an employee table, including salary and benefit information. On the other hand, you don't want one employee to see salary and benefit information for another employee. Employee addresses and phone numbers may not be considered confidential, making it perfectly acceptable for an employee to view someone else's personal information, but not salary information. In other words, a view is an alternative "look" at one or more tables.

To SQL, a view looks like any other table. A view is a database object for which you grant object permissions. See earlier discussion in this chapter. You construct SQL statements to query the view. Because a view is made up of one or more tables, your SQL statement works with views. A view is basically a logical table. In fact, you can do inserts, updates, and deletes on views if they meet certain conditions. Such conditions are that a view must not have calculated fields and the view must include all keys and not null fields for the underlying tables. For more information about SQL, see Chapter 9.

SQL Server 2000 now lets you index views, allowing for greater performance. Prior to SQL Server 2000, views had to rely on the indexes created in the underlying tables. For more information on creating indexes, see Chapter 4 and Chapter 6.

You can create a view in a variety of ways. On the next few pages, I show you all the ways to create a view. Strap yourself in and get going. .

Consider the example that you have a table, named Employees, with the following columns:

EmployeeID

Name

Address

City

State

Zip

Phone

SSN

Hire Date

Terminate Date

Salary

You probably don't want everyone in your organization to know an employee's social security number (SSN), salary, address, and phone number. You can't just leave them out of the table because they must exist for the corporate records. You also don't want to create another table just to store the data in the few columns that everyone in the organization can see. If you do create another table, you are storing the same data in multiple places. This is a bad thing!

The solution is to construct a view, called Emp_View, that hides these columns. The Emp_View view then contains these columns:

EmployeeID

Name

Hire Date

Terminate Date

The view references the Employees table so that data is not stored in multiple places, but it also doesn't force you to return data that everyone can see. Therefore, the term **view** is used to describe an alternative way to look at the Employees table.

After a view is created, you access it just as you access a table. You can construct queries against it. You can also insert and update data into a view. For all intents and purposes, a view **looks** just like a table.

You can create a view in one of three ways. You can use the Create View Wizard, SQL Server Enterprise Manager, or Transact-SQL. The next few pages outline how to use all three methods.

Creating a view with the Create View Wizard

The Create View Wizard, one of many wizards included with SQL Server 2000, guides you through a series of steps, prompting you for input along the way (see Appendix A for a flowchart of the steps). To create a view by using the Create View Wizard, follow these steps:

1. Choose Start-Programs-Microsoft SQL Server-Enterprise Manager to start the SQL Server Enterprise Manager (refer back to Figure BC2-2.)

2. Expand the tree so that you see the name of your server.

For more information about expanding the tree, see the Introduction of this book.

3. Bring up the Select Wizard dialog box by choosing Tools-Wizards.

The Select Wizard dialog box appears (refer back to Figure BC2-3).

4. Select the Create View Wizard item (under the Database category) with the left mouse button and click OK.

The Create View Wizard prompts you to create your view by asking you a series of questions. These questions are presented in steps. The first step is an introduction dialog box letting you know what the wizard will do for you. Click the Next button when you are ready to begin.

5. For the **Select a Database** step that appears, specify the name of the database and click **Next**.

Select from the drop-down list of existing databases for the Database Name field. This database is used to store the view and to provide tables for the view.

6. In the **Select Objects** dialog box that appears (shown in Figure BC2-12), specify the object(s) to be used in the view and then click **Next**.



Figure BC2-12: Select your tables to be referenced in the view.

Click the row in the grid shown in the **Include in View** column for each table to be included in the view. For the example I gave at the beginning of this section, I click **Employees**.

7. In the **Select Columns** dialog box (shown in Figure BC2-13), specify the column(s) to be used in the view. Then click **Next**.



Figure BC2-13: Tell SQL Server which columns you want to display in the view.

Click the row in the grid shown in the **Select Column** column for each column in the tables chosen in the prior step that are to be included in the view. For the example I gave at the beginning of this section, I click **EmployeeID**, **Name**, **Hire Date**, and **Terminate Date**.

8. In the **Define Restriction** dialog box (shown in Figure BC2-14), specify additional criteria for the view. (However, for your first view, leave this step blank for simplicity's sake and click **Next** to continue.)

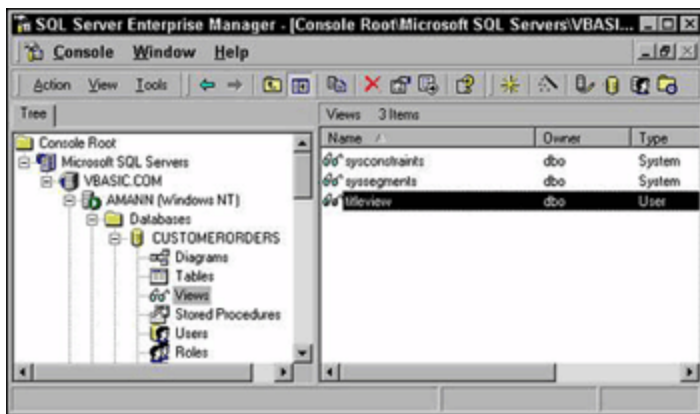


Figure BC2-14: Define how you want to limit the number of rows returned in the view.

Additional criteria, also known as a **restriction**, can be specified to limit the number of records that are returned in the view. This field is optional. This limit is any valid WHERE clause (including the WHERE keyword). For more information about SQL or a WHERE clause, see Chapter 9. If you want all records to be returned, leave this step blank.

The criteria that you specify in this step can also be used to join columns in tables. For example, if you wanted to join the EMPLOYEES table with the ORDERS table, based on the EmployeeID column in each table, you would enter:

```
WHERE EMPLOYEES.EmployeeID = ORDERS.EmployeeID
```

9. In the Name the View dialog box, name the view and then click Next.

Because all SQL Server objects need a name, this step is used to specify that name. By default, you are presented a name that is the first table chosen, followed by _VIEW. This is a good convention to use, but some organizations prefer that the identifier for the type of object be specified before the object name, such as vw_Employees. Simply accept this default.

10. Review your choices in the Completing the Create View Wizard dialog box. If you like what you see and it's exactly what you want, click Finish.

To complete the Create View Wizard, review the SQL that creates the view. This SQL is generated by answering the questions presented by the steps in the Create View Wizard.

If you want to change any of the criteria that you specified in an earlier step, you can do so by clicking the Back button until you reach the desired step to change.

Creating a view with the SQL Server Enterprise Manager

Using the Create View Wizard is a bit easier, but Microsoft also includes a more graphical way to create views. Besides the Create View Wizard, you can use the SQL Server Enterprise Manager to create views. You may want to use the Enterprise Manager rather than the Create View Wizard if you want to create all your database objects the same way. After all, the Enterprise Manager is supposed to give you a single point of entry to manage your enterprise. To use the SQL Server Enterprise Manager for creating a view, follow these steps.

1. Choose Start-Programs-Microsoft SQL Server-Enterprise Manager to start the SQL Server Enterprise Manager.

2. Expand the tree so that you see the Databases folder, then the database in which you want to create the view, and finally the Views folder.

For more information about expanding the tree, see the Introduction of this book.

3. Highlight the Views folder.

After highlighting the Views folder, notice the list of views on the right part of the screen (see Figure BC2-15).

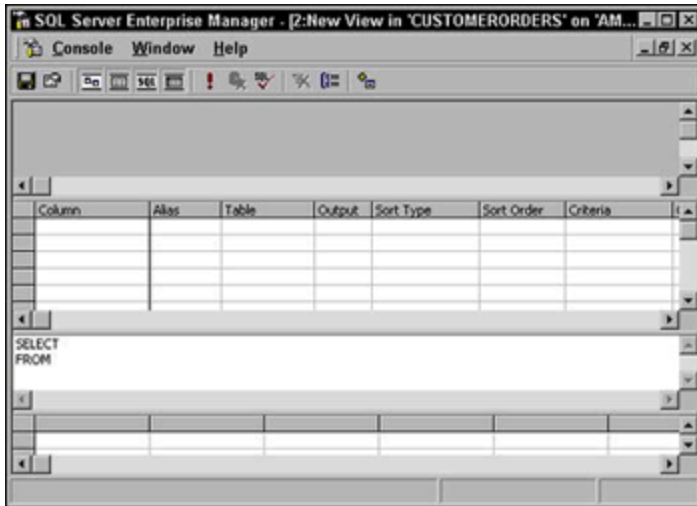


Figure BC2-15: The Enterprise Manager showing the Views folder.

4. Select Action-New View to bring up the New View screen within the Enterprise Manager.

Alternatively, you can use the mouse to right-click anywhere on the right part of the screen and select the New View menu. Also, you can right-click the Views folder in the tree and select the New View menu.

Each method brings up a screen, enabling you to create your new view. This screen is created within the context of the Microsoft Management Console (MMC), more commonly known as the SQL Server Enterprise Manager.

Because multiple windows are open in the Enterprise Manager, you can switch between the open windows by choosing the Window menu. The drop-down list presents the names of all open windows -- take your pick! Choose the desired open window.

Creating the view is done in a series of logical steps. These steps are not actually listed on the screen. I have broken the process into these logical steps because this is one of the few tools in SQL Server 2000 that is not very intuitive to use.

5. Indicate which table(s) to use in your view by entering a SQL statement.

To determine which table(s) to use in your view is sometimes no easy task. I can't really tell you, either. All I can mention is that you need to logically decide which tables store the data to be included in the view. If the data resides in more than one table, you need to decide which columns in those tables are used to join the two tables together. For more information about joins, see Chapters 4 and 9.

It isn't very intuitive, but the SQL pane shown in the middle of the screen is used to generate a graphic in the Diagram Pane, showing all the tables entered in the diagram pane. (Refer to Figure BC2-17 for the location of these panes.) Type the table names into your SQL statement (in the SQL Pane) using this syntax:

```
SELECT *  
FROM table1[,table2][,n...]
```

Substitute the name(s) of your tables for table1 and table2, and so on. For example, I create a simple view and use the table Employees. Therefore, I enter:

```
SELECT *  
FROM Employees
```

6. Execute the SQL statement by clicking the exclamation point icon on the toolbar.

A graphical representation of the tables appears, showing all columns in the diagram pane (see Figure BC2-16).

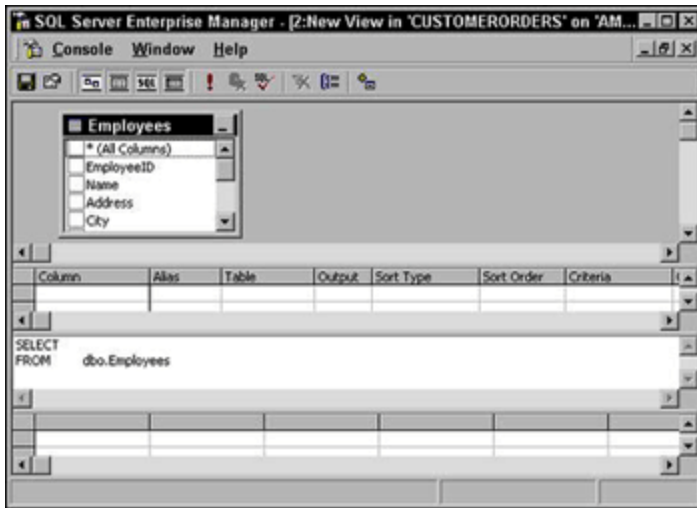


Figure BC2-16: The New View screen showing tables graphically in the diagram pane.

For more information about SQL, see Chapter 9.

7. In the table shown in the diagram pane, check any columns that you want to expose in your view; build your query by checking columns to expose.

Any columns that you want to expose in your view need to be chosen, or checked, in the respective tables shown in the diagram pane. As you click a column, notice that the query is being built dynamically in the SQL pane as you make your choices.

If you have followed my example earlier in this chapter and you want to make only the EmployeeID, Name, Hire Date, and Terminate Date available (exposed) in the view, click these columns (see Figure BC2-17).

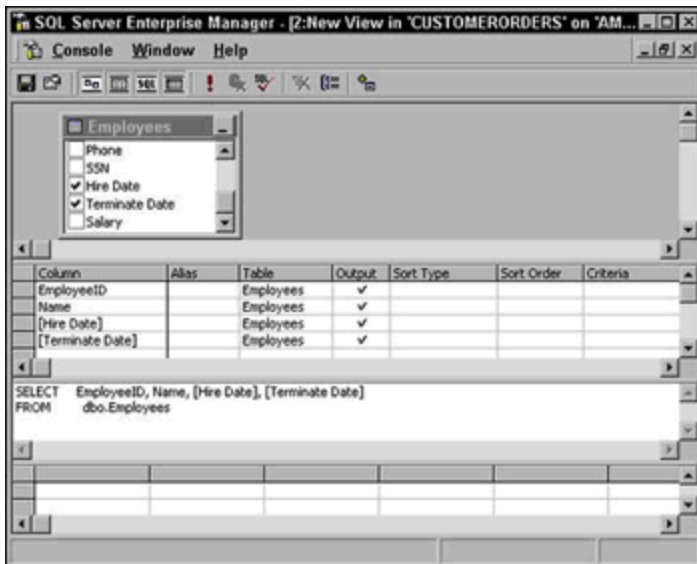


Figure BC2-17: The New View screen after you choose columns to expose in your view.

8. Review your view. (Sounds like a Schoolhouse Rock episode!)

Check the results pane to make sure that your view looks the way you expect. Also, check the names of your columns. When you check columns that you want to expose in your view in Step 7, the system automatically gives your columns alternate names -- or an **alias** to your columns.

To **alias** your columns is to give your columns alternate names. Your SQL statements then refer to these alternate names. Therefore, it's important that your columns are not "aliased" if you don't want them to be.

If you want your columns to be named the same as the actual column names in the real underlying tables, simply click the grid in the Alias column for the desired row in the grid pane. Then clear out the alias text by pressing the backspace key until the line is completely clear. Notice that the query is being built dynamically in the SQL pane as you make your choices.

Also, by default, the wildcard * is shown in the first row in the grid pane. If you don't want to see every column, you need to either deselect this row or completely delete the row. To deselect the row, ensure that there is no check mark in the Output column of this row. If a check mark is there, click the check mark to deselect it. To remove the row, right-click the leftmost column in the desired row and then click the right mouse button and choose the Delete menu option.

If you have any errors to correct, make your changes and re-execute the query.

9. Save your view by clicking the save icon.

You are prompted to enter a name for the view. Make the most of this creative opportunity -- this is the name that you'll use when you create SQL statements, so go for a name that is something meaningful. For example, if you're creating a view for the Human Resources Department, consider a name such as vw_personnel.

Creating a view with SQL

By using the SQL Server Query Analyzer, you can issue SQL statements directly to the server to create a new view. I've seen many people who like to do things the "old-fashioned" way -- they code it. This section is for those people.

To issue the SQL statement to create a view, follow these steps:

1. Choose Start-Programs-Microsoft SQL Server-Query Analyzer to start the SQL Server Query Analyzer.

2. Type the SQL needed to create a view.

You create a view with SQL by issuing the CREATE VIEW statement in the Query Analyzer. For more information about the CREATE VIEW statement, see Chapter 9.

Here's how you can create a view named EMPLOYEE_VIEW:

```
CREATE VIEW EMPLOYEES_VIEW
```

```
AS
```

```
SELECT EmployeeID, Name, [Hire Date], [Terminate Date]
```

```
FROM EMPLOYEES
```

In the sample CREATE VIEW statement, I show the use of square brackets on some of the column names and not on others. Why? The answer is that the two columns, Hire Date and Terminate Date, all contain spaces in the names. If the brackets were not used, SQL Server would have no idea what the column names are.

Everything in the CREATE VIEW statement after the AS clause is a regular Transact-SQL Statement that returns results when you execute the view. Therefore, I recommend that if you create your views by using the CREATE VIEW SQL statement, issue the statement after the AS clause by itself to see whether you get the results you expect. If you don't, you certainly don't want to create a view using these results. For example, issue this statement:

```
SELECT EmployeeID, Name, [Hire Date], [Terminate Date]
```

FROM EMPLOYEES

If the results are what you expect, you can use this statement within the CREATE VIEW statement. If the results are not what you expect, alter the statement. Issuing SELECT queries by themselves in the Query Analyzer is easier than recreating your view object if your Transact-SQL statement for the view is incorrect. For information on using the Query Analyzer, see Chapter 10.

3. Execute your query in the Query Analyzer.

Again, the Query Analyzer is covered in Chapter 10.

4. Test your view with the SELECT statement and execute the statement.

After your view is created, you can test it by using the Query Analyzer, like this:

```
SELECT * FROM EMPLOYEES_VIEW
```