# **Overview of Installing SQL Server 2000**

Microsoft® SQL Server<sup>™</sup> 2000 Setup creates a new installation of SQL Server 2000 or upgrades an earlier version. Before installing or upgrading to SQL Server 2000 it is recommended that you review the following topics.

Торіс	Description
Preparing to Install SQL Server	Lists steps to take before running SQL
2000	Server 2000 Setup
SQL Server 2000: Editions and	Presents an overview of the editions of
<u>Components</u>	SQL Server 2000, installation options,
	and components
Upgrading an Existing Installation	Outlines options for upgrading to SQL
of SQL Server	Server 2000 from an earlier version

For more information, see related topics in the table below.

To install	See
SQL Server 2000 (typical	How to install SQL Server 2000 (Setup)
installation of the relational	
database engine, client tools, and	
client connectivity components)	
SQL Server 2000 client tools only	How to install client tools only (Setup)
(includes management tools and	
client connectivity components,	
no server required)	
SQL Server 2000 connectivity	How to install connectivity only (Setup)
only	
(installs only the client	
connectivity components, no other	
options)	
A named instance or multiple	How to install a named instance of SQL
instances of SQL Server 2000	<u>Server 2000 (Setup)</u>
A SQL Server 2000 virtual server	Before Installing Failover Clustering
for failover clustering	

Analysis Services	Installing Analysis Services
English Query	Installing English Query

## **Preparing to Install SQL Server 2000**

Before installing Microsoft<sup>®</sup> SQL Server<sup>™</sup> 2000, consider the following:

- Be sure the computer meets the system requirements for Microsoft SQL Server 2000. For more information, see <u>Hardware and Software</u> <u>Requirements for Installing SQL Server 2000</u>.
- Back up your current installation of Microsoft SQL Server if installing SQL Server 2000 on the same computer.
- If installing a failover cluster, disable NetBIOS on all private network cards before running SQL Server Setup. For more information, see <u>Before Installing Failover Clustering</u>.
- Review all SQL Server installation options and be prepared to make the appropriate selections when running Setup. For more information about editions and components, see <u>SQL Server 2000: Editions and</u> <u>Components</u>.
- If you plan to install SQL Server to a location other than the default file locations, see <u>File Paths for SQL Server 2000</u>.
- If using an operating system with Regional settings other than English (United States), or if customizing character set or sort order settings, review topics on collation settings. For more information, see <u>Collation</u> <u>Options for International Support</u>.

#### **Before Running SQL Server 2000 Setup**

Before running Setup:

• Create one or more domain user accounts if installing SQL Server 2000 on a computer running Microsoft Windows NT® or Microsoft

Windows® 2000, and you want SQL Server 2000 to communicate with other clients and servers. For more information, see <u>Creating Security</u> <u>Accounts</u>.

- Log on to the operating system under a user account that has local administrative permissions, or assign the appropriate permissions to the domain user account.
- Shut down all services dependent on SQL Server. This includes any service using ODBC, such as Microsoft Internet Information Services (IIS).
- Shut down Microsoft Windows NT Event Viewer and registry viewers (Regedit.exe or Regedt32.exe).

# Hardware and Software Requirements for Installing SQL Server 2000

The minimum hardware and software requirements for running Microsoft® SQL Server<sup>™</sup> 2000 are listed in the following tables.

## **Hardware Requirements**

This table shows hardware requirements for installing Microsoft SQL Server 2000 or SQL Server client management tools and libraries.

Hardware	Minimum requirements
Computer	Intel® or compatible
	Pentium 166 MHz or higher.
Memory (RAM) <sup>1</sup>	Enterprise Edition: 64 MB minimum, 128 MB or more recommended
	Standard Edition: 64 MB minimum
	Personal Edition: 64 MB minimum on Windows 2000, 32 MB minimum on all other operating systems
	Developer Edition: 64 MB minimum
	Desktop Engine: 64 MB minimum on Windows 2000, 32 MB minimum on all other operating systems
Hard disk space <sup>2</sup>	SQL Server database components: 95 to 270 MB, 250 MB typical
	Analysis Services: 50 MB minimum, 130 MB typical
	English Query: 80 MB
	Desktop Engine only: 44 MB

Monitor	VGA or higher resolution
	800x600 or higher resolution required for the SQL Server graphical tools
Pointing device	Microsoft Mouse or compatible
CD DOLLL	

CADdROMndfilling may be reRequired inding on operating system requirements.

2 Actual requirements will vary based on your system configuration and the applications and features you choose to install.

**Note** Microsoft SQL Server 2000 does not have a hardware compatibility list (HCL). If your computer meets the minimum requirements listed in the preceding table, SQL Server 2000 software works on the hardware certified for use with the Microsoft Windows® operating system. For more information about hardware certified for use with the Windows operating system, see the Microsoft Windows Hardware Compatibility List at <u>Microsoft Web site</u>.

#### **Operating System Requirements**

This table shows the operating systems that must be installed to use the various
editions or components of Microsoft SQL Server 2000.

SQL Server edition	
or component	Operating system requirement
Enterprise Edition	Microsoft Windows NT Server 4.0, Microsoft Windows NT Server Enterprise Edition 4.0, Windows 2000 Server, Windows 2000 Advanced Server, and Windows 2000 Data Center Server. Note that Microsoft Windows 2000 Server (any version) is required for some SQL Server 2000 features.
Standard Edition	Microsoft Windows NT Server 4.0, Windows 2000 Server, Microsoft Windows NT Server Enterprise Edition, Windows 2000 Advanced Server, and Windows 2000 Data Center Server.
Personal Edition	Microsoft Windows Me, Windows 98, Windows NT Workstation 4.0, Windows 2000 Professional, Microsoft Windows NT Server 4.0, Windows 2000

	Server, and all the more advanced Windows operating systems.
Developer Edition	Microsoft Windows NT Workstation 4.0, Windows 2000 Professional, and all other Windows NT and Windows 2000 operating systems.
Client Tools Only	Microsoft Windows NT 4.0, Windows 2000 (all versions), Windows Me, and Windows 98.
Connectivity Only	Microsoft Windows NT 4.0, Windows 2000 (all versions), Windows Me, Windows 98, and Windows 95.

**Note** Microsoft Windows NT® Server 4.0, Service Pack 5 (SP5) or later must be installed as a minimum requirement for all SQL Server 2000 editions.

SQL Server 2000 is not supported on Windows NT 4.0 Terminal Server.

For installations of SQL Server 2000 Personal Edition on Windows 98 computers without a network card, Windows 98 Second Edition is required.

#### **Internet Requirements**

This table shows Internet requirements related to using Microsoft SQL Server 2000.

Component	Requirement	
Internet software	Microsoft Internet Explorer 5.0 is required for all installations of Microsoft SQL Server 2000, as it is required for Microsoft Management Console (MMC) and HTML Help. A minimal install is sufficient, and Internet Explorer is not required to be the default browser.	
	Exception to the Internet Explorer 5.0 requirement: If using the Connectivity Only option and not connecting to a server that requires encryption, Microsoft Internet Explorer 4.01 with Service Pack 2 is sufficient.	

Internet Information	If writing XML applications, see <u>System</u>
Services	<b>Requirements for the IIS Virtual Directory</b>
	Management for SQL Server Utility.

#### **Network Software Requirements**

Microsoft Windows NT, Windows 2000, Windows Me, Windows 98, and Windows 95 have built-in network software. Additional network software is required only if you are using Banyan VINES or AppleTalk ADSP. Novel NetWare IPX/SPX client support is provided by the NWLink protocol of Windows Networking.

**Note** TCP/IP must be enabled at the operating system level before installing SQL Server 2000. For more information, see <u>Network Libraries</u>.

#### **Supported Clients**

Microsoft SQL Server 2000 supports the following clients: Windows NT Workstation, Windows 2000 Professional, Windows 98, Windows 95, Apple Macintosh®, OS/2, and UNIX. Macintosh, OS/2, and UNIX do not support the SQL Server graphical tools and require ODBC client software from a third-party vendor.

## **Considerations for Other Microsoft Products**

The following Microsoft products require Service Release or Service Packs to operate correctly with SQL Server 2000.

## Access 2000

Microsoft Access 2000 requires the installation of either Microsoft Office 2000 Service Release 1 (SR1) or Access 2000 SR1 to operate correctly with SQL Server 2000. If running an earlier version of Access 2000, you cannot test automatic data processing (ADP) applications against SQL Server 2000. You cannot access database diagrams, stored procedures, table designs, or view designs. Other issues to be addressed in a future Access Service Release:

- When you run Access 2000 with SR1, you can test ADP applications. You can also alter database diagrams, stored procedures, table designs, or view designs, but you cannot save any changes. A future Access Service Release will allow limited ability to save changes.
- The Access 2000 Create Database Wizard cannot successfully create a SQL Server 2000 database. You can work around this by first creating the database using SQL Server Enterprise Manager, and then creating an ADP for the database using the **Project (Existing Database)** option on the **New** dialog box in Access 2000.
- The Access 2000 Upsizing Wizard does not support upsizing to SQL Server 2000. You can work around this by using Data Transformation Services in the Enterprise Manager to import your MDB database file into SQL Server. You can then rename your MDB tables and create linked tables to the resulting SQL Server database with the same names as your original MDB table names.

## Visual Studio 6.0

When you run Microsoft Visual Studio® 6.0, you cannot access database diagrams, stored procedures, table designs, or view designs in SQL Server 2000. Visual Studio 6.0 Service Pack 4 allows you to alter database diagrams, stored procedures, table designs, or view designs, but you cannot save them. A future Visual Studio Service Pack will allow a limited ability to save changes.

The SQL Server 2000 tools cannot access database diagrams saved using the design tools in Visual Studio 6.0 until you have modified the **dtproperties** table in the database. For more information, see <u>Backward Compatibility</u>.

#### See Also

Editions of SQL Server 2000

SQL Server 2000: Editions and Components

Operating Systems Supported by the Editions of SQL Server 2000

# SQL Server 2000: Editions and Components

SQL Server 2000 editions include the Enterprise Edition, the Standard Edition, the Personal Edition, the Developer Edition, and the Evaluation Edition.

**Note** The Microsoft SQL Server 2000 Personal Edition replaces the Microsoft SQL Server version 7.0 Desktop Edition. To install client tools only, you can use the Personal Edition, which is also available when you purchase the Standard and Enterprise Editions of SQL Server 2000.

When you select **SQL Server 2000 Components** on the opening screen, three options appear on the **Install Components** screen:

#### **Install Database Server**

Starts SQL Server Setup, with screens for selecting installation options.

#### **Install Analysis Services**

Installs Analysis Services on computers processing OLAP cubes. For more information, see <u>Installing Analysis Services</u>.

#### **Install English Query**

Installs English Query on computers running English Query applications. For more information, see <u>Installing English Query</u>.

#### **Choosing Components and Options to Install**

You may have a database server, an Internet server, or require a database on a client computer. If running database client/server applications you may or may not require a database on your computer. You may need tools to administer a database server, or you may want to run applications that access an instance of SQL Server. Installation choices for these and other SQL Server configurations are described in the following paragraphs.

#### **Installing SQL Server on a Database Server**

If installing a database server, install either SQL Server 2000 Enterprise Edition or SQL Server 2000 Standard Edition. If installing a personal database on your

workstation, install SQL Server 2000 Personal Edition. These installations typically include the database engine, the client database management tools, and the client connectivity components.

On a database server, you can install a default instance of SQL Server 2000 relational database engine. You can also install one or more named instances of the SQL Server 2000 database engine. Other than specifying an instance name, the setup choices are similar to those for installing a default instance.

When installing an instance of SQL Server 2000, you must specify whether you want the instance to use failover clustering. For more information, see <u>Before</u> <u>Installing Failover Clustering</u>.

## **Using SQL Server with Client/Server Applications**

For a computer running database client/server applications, such as Microsoft Visual Basic® applications that connect directly to an instance of SQL Server, you have several options:

- If you require a personal database on your client computer, install the Personal Edition of SQL Server. This setup typically installs the client tools and client connectivity components along with the database engine.
- If you do not require a database on your computer, but need to administer an instance of SQL Server on a database server, or plan to develop SQL Server applications, install the option for Client Tools Only. This option includes the client connectivity components. For more information, see <u>How to install client tools only (Setup)</u>.
- If you want to only run applications that access instances of SQL Server on database servers, install the connectivity only components. For more information, see <u>How to install connectivity only (Setup)</u>.

#### Using SQL Server with an Internet Server

On an Internet server, such as a server running Microsoft Internet Information Services (IIS), you typically install the SQL Server 2000 client tools. Client tools

include the client connectivity components used by an application connecting to an instance of SQL Server. In addition, the client tools include the utility for configuring the virtual roots needed for applications to access SQL Server through URLs.

After installing the SQL Server client tools, you configure the virtual roots that support accessing an instance of SQL Server through a URL. For more information about configuring the virtual roots, see <u>Using IIS Virtual Directory</u> <u>Management for SQL Server Utility</u>.

**Note** Although you can install an instance of SQL Server on a computer running IIS, this is typically done only for small Web sites that have a single server computer. Most Web sites have their middle-tier IIS system on one server or cluster of servers, and their databases on a separate server or federation of servers. For more information about federations, see Federated SQL Server 2000 Servers.

If some of the Web pages on an Internet server use English Query, you would also install that component.

## **Other SQL Server Components**

• For distributing SQL Server 2000 with applications, use the SQL Server 2000 Desktop Engine, a stand-alone database engine that independent software vendors can package with their applications. For more information, see <u>Distributing SQL Server with Applications</u>.

**Note** The Desktop Engine has no graphical user interface and is not related to the SQL Server 7.0 Desktop Edition.

In addition to the major components and editions shown on the **Install SQL Server 2000 Components** screen, other editions of SQL Server 2000 are available: SQL Server 2000 Developer Edition, SQL Server 2000 Windows CE Edition, and the SQL Server 2000 Enterprise Evaluation Edition. For more information, see <u>Features Supported by the Editions of SQL Server 2000</u>.

## See Also

Editions of SQL Server 2000

Operating Systems Supported by the Editions of SQL Server 2000

# **Installing English Query**

English Query is a development tool that works with Microsoft® SQL Server<sup>™</sup> 2000. Using English Query, you can create applications that allow users to query a SQL Server database or an Analysis Services database in English. For example, users can ask, "How many widgets were sold in Washington last year?" instead of using the SQL statements:

SELECT sum(Orders.Quantity) FROM Orders, Parts WHERE Orders.State='WA' AND Datepart(Orders.Purchase\_Date,'Year')='1999' AND Parts.PartName='widget' AND Orders.Part\_ID=Parts.Part\_ID

When you install English Query, **English Query** is added to the **Microsoft SQL Server** program group on the **Start** menu. **English Query** contains these shortcuts:

- English Query Books Online
- English Query Tutorials
- Microsoft English Query

If English Query is not installed with Microsoft SQL Server 2000, the Help system will access English Query Books Online, Eqdoc.chm, instead of SQL Server Books Online, SQL80.col. However, both documentation files contain essentially the same material about English Query and both provide contextsensitive (F1) Help for English Query. Regardless of the installation scenario, English Query Books Online is available from the English Query program group.

Installation Requirements for English Query are:

• Microsoft Windows® 95, Microsoft Windows 98, or Microsoft Windows NT® version 4.0 or later

- 40 MB of free disk space
- Microsoft Internet Explorer 5.0 or later

## To install English Query

# **Installing Analysis Services**

Microsoft<sup>®</sup> SQL Server<sup>™</sup> 2000 Analysis Services includes a powerful server for the construction and analysis of multidimensional data.

#### **To install Analysis Services**

# **Setting up Windows Services Accounts**

On the Microsoft® Windows NT® and Microsoft Windows® 2000 operating systems, Microsoft SQL Server<sup>™</sup> and SQL Server Agent are started and run as Windows services. These services appear in the list of installed services in the Services dialog box, available using Windows Control Panel. The table shows each service name and the term used to refer to the default and named instances of SQL Server, as displayed in the Services dialog box.

Service		Term for default	Term for named
	Name	instance	instance
Microsoft	SQL	MSSQLSERVER	MSSQ\$InstanceName
SQL Server	Server		
Microsoft	SQL	SQLSERVERAGENT	SQLAgent\$InstanceName
SQL Server	Server		
Agent	Agent		

For Microsoft SQL Server<sup>™</sup> and SQL Server Agent to run as services in Windows, they must be assigned a Windows user account. Typically, both SQL Server and SQL Server Agent are assigned the same user account, either the local system or domain user account. However, you can customize the settings for each service during the installation process. For more information about how to customize account information for each service, see <u>Services Accounts</u>.

**Note** Microsoft Windows 98 does not support Windows services; instead, SQL Server simulates the SQL Server and SQL Server Agent services. It is not required that you create user accounts for these simulated services.

#### **Using the Local System Account**

The local system account does not require a password, does not have network access rights in Windows NT 4.0 and Windows 2000, and restricts your SQL Server installation from interacting with other servers.

## Using a Domain User Account

A domain user account uses Windows Authentication, that is, the same user name and password used to connect to the operating system is also used to connect to SQL Server. A domain user account is typically used because many server-to-server activities can be performed only with a domain user account, for example:

- Remote procedure calls.
- Replication.
- Backing up to network drives.
- Heterogeneous joins that involve remote data sources.
- SQL Server Agent mail features and SQL Mail. This restriction applies if using Microsoft Exchange. Most other mail systems also require clients (the SQL Server and SQL Server Agent services) to be run on accounts with network access.

**Note** Several servers running SQL Server can share the same user account. When setting up replication, it is recommended that a Publisher and all its Subscribers share the same service account for the SQL Server service.

## **Requirements for Domain User Account**

All domain user accounts must have permission to:

- Access and change the SQL Server directory (\Program Files\Microsoft SQL Server\Mssql).
- Access and change the .mdf, .ndf, and .ldf database files.
- Log on as a service.

- Read and write registry keys at and under:
  - HKEY\_LOCAL\_MACHINE\Software\Microsoft\MSSQLServe -or- for any named instance: HKEY\_LOCAL\_MACHINE\Software\Microsoft\Microsoft SQL Server.
  - HKEY\_LOCAL\_MACHINE\System\CurrentControlset\Service
     -or- for any named instance:
     HKEY\_LOCAL\_MACHINE\System\CurrentControlset\Service
     HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows
     NT\CurrentVersion\Perflib.

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

Service	Permission	Functionality
SQL Server	Network write privileges	Write to a mail slot using
		xp_sendmail.
SQL Server	Act as part of operating	Run <b>xp_cmdshell</b> for a user
	system and replace a	other than a SQL Server
	process level token	administrator.
SQL Server	Member of the	Create CmdExec and
Agent	Administrators local group	ActiveScript jobs belonging
		to someone other than a SQL
		Server administrator.
		Use the autorestart feature.
		Use run-when-idle jobs.
SQL Server	Member of local Power	Add and delete SQL Server
	Users or local	objects in the Windows 2000
	Administrators group	Active Directory.

This table shows additional permissions required for certain functionality.

#### **Changing User Accounts**

To change the password or other properties of any SQL Server–related service after installing SQL Server, use SQL Server Enterprise Manager. If your Windows password expires and you change it, be sure to also revise the SQL Server services settings in Windows. For more information, see <u>Changing Passwords and User Accounts</u>.

#### See Also

Creating Security Accounts
Planning Security
Services Accounts

# File Paths for SQL Server 2000

In Microsoft® SQL Server<sup>™</sup> 2000, the default location for the installed SQL Server files has changed. For the default instance of SQL Server, the default directory for both program and data files is \Program Files\Microsoft SQL Server\Mssql. You can specify a file path other than the default for both program and data files.

Shared Tools are installed by default at \Program Files\Microsoft SQL Server\80\Tools. This folder contains files shared by all instances of SQL Server 2000, both default and named. Tools include SQL Server Books Online, Dev Tools, and other components.

Setup also installs files in the Microsoft Windows® system directory. The system file location cannot be changed.

#### **SQL Server Program File Location**

The SQL Server program files are located in \Program Files\Microsoft SQL Server\Mssql\Binn.

The program file location is the root directory where Setup creates the folders that contain program files and files that typically do not change as you use SQL Server. Although these files are not read-only, the folders do not contain data, log, backup files, or replication data; therefore, the space requirements for these files should not increase as SQL Server is used.

**Note** Program files cannot be installed on a removable disk drive.

#### **SQL Server Data File Location**

The SQL Server data files are located in \Program Files\Microsoft SQL Server\Mssql\Data.

The data file location is the root directory where Setup creates the folders that contain database and log files, as well as directories for the system log, backup, and replication data. Setup creates database and log files for the **master**, **model**, **tempdb**, **msdb**, **pubs**, and **Northwind** databases. The SQL Server data file path should be located on a drive that has space available for these files to grow.

**Note** Data files cannot be installed on a file system using compression.

## **Specifying File Paths**

In SQL Server 2000, due to multiple instance options, the instance name is used in addition to the user-specified location for program and data files. For tools and other shared files, however, instance names are not required.

## **Default Instance File Path for Program and Data Files**

For the default instance of SQL Server, the default SQL Server directory name (Mssql) is used as the default instance name, along with the directory you specify.

For example, if you specify that the SQL Server default instance be installed at D:\MySqlDir, the file paths are:

- D:\MySqlDir\Mssql\Binn (for program files)
- D:\MySqlDir\Mssql\Data (for data files)

## Named Instance File Path for Program and Data Files

For any named instances, the given name of the instance is used with the directory specified.

For example, if you specify that the instance named **MyInstanceA** be installed at D:\MySqlDir, the paths are:

- D:\MySqlDir\MSSQL\$MyInstanceA\Binn (for program files)
- D:\MySqlDir\MSSQL\$MyInstanceA\Data (for data files)

## See Also

File Locations for Multiple Instances of SQL Server

Locating Directories and Files

# **Upgrading an Existing Installation of SQL Server**

You can upgrade from earlier versions to Microsoft® SQL Server<sup>™</sup> 2000, and also perform upgrade operations once SQL Server 2000 is installed. Upgrades to SQL Server 2000 from SQL Server version 6.5 and from SQL Server version 7.0 are different operations. SQL Server 6.5 databases (and related information) are converted to SQL Server 2000 formats. An installation of SQL Server 7.0 is overwritten by SQL Server 2000, unless a named instance configuration is installed, allowing SQL Server 7.0 to remain intact.

After an initial installation of SQL Server 2000, other upgrade options are available. If using more than one instance of SQL Server 2000, you can upgrade one instance by adding components, and have different component sets for multiple instances.

#### **Upgrading from SQL Server 7.0 to SQL Server 2000**

You can overwrite your existing installation of SQL Server 7.0 by installing a default instance of SQL Server 2000. You can also keep your installation of SQL Server 7.0 intact by installing a named instance of SQL Server 2000. Both operations are performed using the following procedure.

#### To upgrade from SQL Server 7.0 to SQL Server 2000

# **Basic Installation Options**

This section describes basic installation options for SQL Server 2000. Upgrading an existing installation, or creating a new installation on either a local or remote computer is considered a basic installation option.

For more information about cluster maintenance, performing an unattended setup, or rebuilding the registry, see <u>Advanced Installation Options</u>.

# **Entering Information in Basic Setup Screens**

Microsoft® SQL Server<sup>™</sup> 2000 Setup provides basic and advanced options. There are two options for a local installation:

- Create a new or additional installation
- Upgrade, remove, or add components to an existing installation

If you choose the upgrade option, you have many other choices available. For more information, see <u>Existing Installation Options</u>. In addition, you can select advanced options. For more information about your initial setup choices, see <u>Installation Selection</u>.

For a basic, local installation, select the option for creating a new or additional installation. After entering user and product identification (ID) information in subsequent screens, choose the components to include in this installation of SQL Server 2000. You can select to install either connectivity only, client tools only (which includes connectivity components), or the complete server and client tools option. For more information, see Installation Definition.

If you choose to install the SQL Server relational database with both server and client tools, select either a named instance or the default instance of SQL Server 2000. For more information, see <u>Instance Name</u>.

After selecting the default instance, or choosing to create a named instance, the standard setup type selection screen is presented. For more information, see <u>Setup Type: Typical, Minimum, or Custom</u>.

Other options may be presented while running Setup, depending on the specifics of your system and installation.

### See Also

Upgrading an Existing Installation of SQL Server

Multiple Instances of SQL Server

# **Computer Name**

The **Computer Name** dialog box in Setup allows you to install Microsoft SQL Server 2000 on your local computer, on a remote computer, or on a virtual server.

All options for installing and upgrading are available on the local computer. Advanced options, including registry rebuild, unattended installation, and upgrading to a cluster are not available on a remote installation. If you are running Setup on a clustered computer, the Virtual Server option is available.

## Options

#### **Local Computer**

By default, the name in the edit box is the local machine name, that is, the computer on which Setup is running. For a local installation, accept the default and click **Next**.

**Note** If you are installing tools only, **Local Computer** will be the only option available on this dialog box.

#### **Remote Computer**

Enter a computer name for a remote installation, or click **Browse** to locate the remote computer.

#### **Virtual Server**

Enter the name of a new or existing Virtual SQL Server to manage.

This option is available only when Microsoft Cluster Service (MSCS) is detected on an Windows NT or Windows 2000 Enterprise operating system.

#### Browse

Click the **Browse** button to locate a remote computer.

This button is available only when the **Remote Computer** option is selected.

### See Also

Installing a Remote Configuration Installing a Virtual Server Configuration Before Installing Failover Clustering Creating a Failover Cluster

# **Installation Selection**

The **Installation Selection** screen is an initial screen in Microsoft® SQL Server<sup>™</sup> Setup, where you select among three options for running the installation program.

## **Options**

#### Create a new instance of SQL Server, or install Client Tools

Creates a new installation of SQL Server 2000; either a default or named instance. In addition, this option allows you to install only client tools using the compact disc for any edition of SQL Server 2000, on any operating system other than Microsoft Windows® 95.

#### Upgrade, remove, or add components to an existing instance of SQL Server

Allows you to upgrade, remove, or add components to an existing instance of SQL Server. Existing instances include installations of earlier versions (SQL Server version 6.5 and SQL Server version 7.0) as well as instances of SQL Server 2000. For more information, see <u>Existing Installation Options</u>.

#### **Advanced Options**

Select advanced options for cluster maintenance, unattended setup, and registry rebuild.

### See Also

Multiple Instances of SQL Server

Upgrading an Existing Installation of SQL Server

**Advanced Installation Options** 

# **Existing Installation Options**

The **Existing Installation Options** Setup screen includes choices for working with upgrades from previous versions of Microsoft® SQL Server<sup>™</sup>, as well as upgrades to SQL Server 2000 components. Options that do not apply to your specific setup do not appear on the screen.

## **Options**

### Add components to your existing installation

Allows you to add components to an existing installation of SQL Server 2000.

#### Uninstall your existing installation

Removes an installation (default or named instance) of SQL Server 2000 from your computer. The instance to remove is specified in the **Instance Name** screen.

#### Upgrade your existing installation

This option is available for use with existing installations of SQL Server 7.0 and SQL Server 2000. Depending on the version, edition, and component makeup of your existing installation, selecting this option starts the process for one of the following upgrades:

- Upgrade from SQL Server 7.0 to SQL Server 2000. (If you cannot upgrade client tools, see Upgrade Issues below.)
- Add components to an existing installation of SQL Server 2000. For example, you may have purchased a SQL Server version with more features, or need to install certain components.

**Note** Upgrades from SQL Server 6.5 to SQL Server 2000 are run using the SQL Server Upgrade Wizard, available on the SQL Server **Start** menu.

### Upgrade your existing installation to a clustered installation

This option is a step in the process of upgrading from a clustered installation of SQL Server 6.5 or SQL Server 7.0 to a clustered installation of SQL Server 2000. First, the earlier version of SQL Server is upgraded to SQL Server 2000. Next, the existing SQL Server 2000 installation can be upgraded to a cluster. For more information, see <u>Upgrading to a SQL Server 2000 Failover Cluster</u>.

## **Upgrade Issues**

- On a computer running SQL Server 7.0 client tools only, you may encounter the following message when you choose to upgrade your existing installation: "The default instance detected is not able to be upgraded. Please select New Install to upgrade your tools." This issue can occur if you have installed the SQL Server 7.0 client tools by choosing **Custom** in the **Setup Type** dialog box, and then by selecting tools in the components dialog box. In this situation, the existing client tools installation of SQL Server 7.0 cannot be upgraded due to registry issues. Instead, you must re-install SQL Server, by selecting **Create a new instance of SQL Server, or install Client Tools**.
- You can upgrade a beta version of SQL Server 2000 to the final version of the product by using the option to upgrade your existing installation. If you are performing such an upgrade on a computer or a cluster containing multiple instances, you must first close all instances of SQL Server before upgrading.

### See Also

Upgrading from SQL Server 7.0 to SQL Server 2000

Upgrading an Existing Installation of SQL Server

Select Components

# **Installation Definition**

Use the **Installation Definition** screen to select the components to include in this installation of SQL Server 2000. If you select Client Tools Only or Connectivity Only, Setup proceeds and no additional choices are required, unless you select components when installing client tools. If you choose to install Server and Client Tools, additional setup screens will appear.

### **Options**

#### **Client Tools Only**

Installs only the client relational database management tools. Included in this option are the client tools for administering SQL Server and the client connectivity components. In addition, this option allows you to select other components to install. For more information, see <u>How to install client tools</u> <u>only (Setup)</u>.

#### Server and Client Tools

Installs both server and client tools to create a relational database server with administrative capabilities. Selecting Server and Client Tools presents the full range of additional setup options.

For more information about performing a typical installation of a default instance of the database engine, including all client and connectivity components, see <u>How to install SQL Server 2000 (Setup</u>).

**Note** This option is not available if you are installing client tools using a compact disc for an edition of SQL Server that is not supported by your computer's operating system.

#### **Connectivity Only**

Installs only the relational database client connectivity components, including MDAC 2.6 (Microsoft Data Access Components), a requirement for connecting to SQL Server 2000 named instances. This option provides connectivity tools only, with no choice of client tools or other components. For more information, see <u>How to install connectivity only (Setup)</u>.

## See Also

Management Tools

Server Components

Client Connectivity

# **User Information**

The **User Information** Setup screen prompts you to supply your name and company name. These fields are required.

When installing on a network, be sure to supply the name of a user responsible for using or administering the server.

### See Also

Setting Up Windows Services Accounts

# **Instance** Name

Use this screen to add and maintain instances of Microsoft® SQL Server<sup>™</sup> 2000.

## **Options**

### Default

- When selected, a default instance of SQL Server 2000 is installed. Click **Next** to proceed with the install process.
- When cleared, you can install or maintain a named instance of SQL Server 2000.

**Note** If this check box is not enabled, Setup has detected a default instance of SQL Server on this computer. The default instance could be an installation of SQL Server 6.5, SQL Server version 7.0, or it could be the default instance of SQL Server 2000, already installed. Only one installation of SQL Server, any version, can be the default instance at any one time. For more information, see <u>Multiple Instances of SQL Server</u>.

#### **Instance** Name

Enter a new instance name, or the name of the instance to maintain. Review and follow the rules for instance names.

**IMPORTANT** It is recommended that instance names be kept to less than 10 characters. Instance names can appear in the user interface of various SQL Server and system tools; shorter names are more readable.

#### **Instance Naming Rules**

- An instance name is not case-sensitive.
- An instance name cannot be the terms *Default* or *MSSQLServer*.

- Instance names must follow the rules for SQL Server identifiers and cannot be reserved keywords.
- Instance names are limited to 16 characters.
- The first character in the instance name must be a letter, an ampersand (&), an underscore (\_), or a number sign (#). Acceptable letters are those defined by the Unicode Standard 2.0, which includes Latin characters a-z and A-Z, in addition to letter characters from other languages.
- Subsequent characters can be:
  - Letters as defined in the Unicode Standard 2.0.
  - Decimal numbers from either Basic Latin or other national scripts.
  - The dollar sign (\$), a number sign (#), or an underscore (\_).
- Embedded spaces or special characters are not allowed in instance names. Neither is the backslash (\), a comma (,), a colon (:), or the at sign (@).

**WARNING** Only characters that are valid in the current Microsoft Windows® code page can be used in instance names in SQL Server 2000. If a Unicode character not supported under the current code page is used, an error occurs.

## See Also

Working with Instances and Versions of SQL Server

Working with Named and Multiple Instances of SQL Server 2000

Naming Conventions for Instances of SQL Server 2000

**Reserved Keywords** 

# Setup Type: Typical, Minimum, or Custom

When you install the Microsoft® SQL Server<sup>™</sup> 2000 Enterprise Edition, SQL Server 2000 Standard Edition, or SQL Server 2000 Personal Edition, SQL Server Setup offers three installation types in the **Setup Type** dialog box. In addition, you can modify the installation location for both program and data files in this dialog box.

### **Options**

#### **Typical**

Installs all of SQL Server using the default installation options. This installation is recommended for most users.

#### Minimum

Installs the minimum configuration necessary to run SQL Server. This installation is recommended for users who have computers with minimum available disk space.

#### Custom

Installs SQL Server and allows you to change any or all of the default options. Use a custom installation to select components and subcomponents, or to change settings for collations, services accounts, authentication, or network libraries.

#### **Destination Folders**

The default installation location is C:\Program Files\Microsoft SQL Server\, for both program and data files.

#### **Program files**

Click **Browse** to select another installation location for the SQL Server program files.

#### **Data files**

Click Browse to select another installation location for the SQL Server data

files.

**CAUTION** It is recommended that program files not be installed on a cluster disk, so that future upgrades to a cluster are possible. If you select a folder on a cluster disk as a destination for SQL Server program files, a message appears requesting another installation path for program files.

When upgrading an installation of SQL Server 7.0 that has previously had program files installed on a cluster disk, a similar message appears: "Setup will move the program files from the cluster disk. Provide a new location for the program files. The drive letter you select must exist on all nodes of the cluster as a local drive so that you can later upgrade to a clustered installation."

For more information, see <u>Upgrading to a SQL Server 2000 Failover Cluster</u>.

## **Components for Each Installation Type**

When you install SQL Server on Microsoft Windows NT<sup>®</sup> or Windows 2000, these options are offered for all installation types:

- Windows Services accounts (logon accounts) for SQL Server and SQL Server Agent.
- Whether to start SQL Server and SQL Server Agent automatically each time the computer is restarted.
- Use of various network libraries, or protocols, including TCP/IP Sockets, Named Pipes, and Multiprotocol.

This table lists the types of installations and components that each installation provides.

Component	Typical	Minimum	Custom
Database Server	Yes	Yes	Optional
Upgrade Tools <sup>1</sup>	Yes	No	Optional
Replication Support	Yes	Yes	Optional
Full-Text Search	Yes	Yes	Optional

Client Management Tools	All	None	Optional
Client Connectivity	Yes	Yes	Not an option
Books Online	Yes	No	Optional
Development Tools	Debugger only	None	Choice of tools
Code Samples	None	None	Choice of samples
Collation Settings	Yes	Yes	Choice of settings

1 Upgrade Tools are installed by default only for the default instance of SQL Server 2000, not for any named instances.

## See Also

Net-Libraries and Network Protocols

# **Select Components**

On the **Select Components** screen, you can choose components and subcomponents to install, or to reinstall if not set up initially. Options for a typical installation are selected by default. Select the components to install or reinstall and clear all others.

**Note** You cannot remove components by clearing check boxes on this screen. The only way to remove installed components is to remove SQL Server entirely.

## **Options**

#### Components

Lists the main components of SQL Server

#### **Sub-components**

Lists the sub-components available for the selected component

SQL Server components and respective subcomponents include:

- <u>Server Components</u>
  - SQL Server
  - Upgrade Tools
  - Replication Support
  - Full-Text Search
  - Debug Symbols
  - Performance Counters
- <u>Management Tools</u>

- Enterprise Manager
- Profiler
- Query Analyzer
- DTC Client Support
- Conflict Viewer
- <u>Client Connectivity</u>
- Books Online
  - Books Online on Disk
- <u>Development Tools</u>
  - Headers and Libraries
  - MDAC SDKs
  - Backup/Restore API
  - Debugger Interface
- <u>Code Samples</u>
  - Choice of many code samples

### See Also

How to add components to an instance of SQL Server 2000 (Setup)

## **Server Components**

These components can be installed from the Server Components category in the **Select Components** dialog box, when running Setup. Server Components are included when the option for Server and Client Tools is selected as an initial installation choice.

#### **SQL Server**

Installs the SQL Server relational database engine and other core tools. If any SQL Server program files are installed, the SQL Server component must be installed.

**Note** When installing the SQL Server component, the Setup program also installs the **bcp**, **isql**, and **osql** utilities, ODBC, OLE DB, and DB-Library.

#### **Upgrade Tools**

Installs the SQL Server Upgrade Wizard, used to upgrade SQL Server 6.5 databases to the current version.

#### **Replication Support**

Installs the scripts and binary files used for replication.

#### **Full-Text Search**

Installs the Microsoft full-text search engine (Microsoft Search service), which extends the ability to search on character columns beyond the basic equality and LIKE operators.

#### **Debug Symbols**

Installs the debug symbols for installations.

#### **Performance Counters**

Installs performance counters for use with installations.

#### See Also

**Installation Definition** 

# **Management Tools**

These components can be installed from the Management Tools category in the **Select Components** dialog box, when running Setup. Management tools are included when the option for Client Tools Only is selected as an initial installation choice.

#### **Enterprise Manager**

Used to perform server and enterprise administrative tasks.

#### Profiler

Used to monitor, record, and support auditing of Microsoft SQL Server database activity.

#### **Query Analyzer**

Used to enter Transact-SQL statements and procedures interactively. Also provides graphical query analysis in the form of graphical showplans.

#### **DTC Client Support**

Used to extend database transactions across multiple servers.

Microsoft Distributed Transaction Coordinator (MS DTC) coordinates transactions across a network of systems running Microsoft Windows NT®, Microsoft Windows® 98, and Microsoft Windows 95.

#### **Conflict Viewer**

Used to view and, if necessary, change the way synchronization conflicts are resolved.

#### See Also

**Installation Definition** 

How to install client tools only (Setup)

# **Client Connectivity**

The client connectivity component is an option in the **Select Components** dialog box in Setup. The client connectivity component is used to communicate between clients and servers, and includes the Microsoft Data Access Components (MDAC) and network libraries for DB-Library, ODBC, and OLE DB.

This component has no subcomponents. Client Connectivity is installed when the option for Connectivity Only is selected as an initial installation choice.

**Note** To connect to a named instance of SQL Server 2000, MDAC 2.6 must be installed on the client computer.

### See Also

**Installation Definition** 

Distributing SQL Server with Applications

# **Books Online**

This component can be installed from the Books Online category in the **Select Components** dialog box in Setup. The Books Online component includes both the full SQL Server Books Online for SQL Server 2000 and online Help, available by clicking the Help button or pressing the F1 key in dialog boxes and interface elements.

#### **Books Online on Disk**

Installs the complete documentation set on your local drive in the default shared tools locations: \Program Files\Microsoft SQL Server\80\Tools\Books.

**Note** You may want to view information in SQL Server Books Online for Microsoft® SQL Server<sup>TM</sup> 7.0. For more information, see <u>How to access SQL</u> <u>Server Books Online for SQL Server 7.0</u>.

# **Development Tools**

These components can be installed from the Development Tools category in the **Select Components** dialog box in Setup. To install development tools, choose a custom installation in the **Setup Type** screen. The Debugger Interface is an exception; it is included when you choose to install a typical installation.

#### **Headers and Libraries**

Installs the include (\*.h) files and library (\*.lib) files needed by a C developer to create programs that use OLE DB, ODBC, DB-Library, Open Data Services, SQL-DMO, Embedded SQL for C, and MS DTC. These files are installed in the \Program Files\Microsoft SQL Server\80\Tools\DevTools\Include and the \...\DevTools\Lib directories by default (shared tools location).

#### **MDAC SDKs**

Installs MDAC and XML Software Development Kits.

#### **Backup/Restore API**

Installs the header files, sample programs, and documentation required by software vendors to develop custom applications to back up and restore Microsoft SQL Server databases.

#### **Debugger Interface**

Installs an interface for stored procedure debugging.

# **Code Samples**

The samples component is available from the Code Samples category in the **Select Components** dialog box in Setup. This component installs programming sample files used for reference when you write programs for Microsoft® SQL Server<sup>™</sup> 2000. These files are installed in folders in the \Program Files\Microsoft SQL Server\80\Tools\DevTools\Samples directory by default (shared tools location). You can install any or all of these samples.

**Note** Sample code for using a virtual device to backup or restore data is included in the Backup/Restore API, a sub-component of the Development Tools component. For more information, see <u>Development Tools</u>.

Option	Name	
ADO	Microsoft ActiveX <sup>®</sup> Data Objects	
DBLIB	DB-Library	
Desktop	Desktop Engine	
DTS	Data Transformation Services	
ESQLC	Embedded SQL for C	
Misc	Miscellaneous Samples	
MSDTC	Microsoft Distributed Transaction Coordinator	
ODBC	Open Database Connectivity	
ODS	Open Data Services	
OLE Automation	OLE Automation	
Replication	Replication	
Silver	Sample Database Schemas	
SQL-DMO	SQL Distributed Management Objects	
SQL-NS	SQL Namespace	
Utils	Sample Utilities	
XML	XML Samples	

# **Network Libraries**

On the **Network Libraries** screen, you can select network libraries to install for Microsoft® SQL Server<sup>™</sup> 2000. Network libraries are used to pass network packets between clients and a server running SQL Server. The network libraries, implemented as dynamic-link libraries (DLLs), perform the network operations required to communicate using specific interprocess communication (IPC) mechanisms.

A server can listen on, or monitor, multiple network libraries at one time. During installation, SQL Server Setup installs all of the Net-Libraries onto the computer and allows you to configure some or all of the Net-Libraries. If a particular Net-Library is not configured, the server cannot listen on that Net-Library. After installation, you can change these configurations using the Server Network utility.

For a clustered installation, only Named Pipes and TCP/IP are available. When installing a clustered instance, the unsupported network libraries are unavailable. When you install named instances, the Multiprotocol, AppleTalk, and Banyan VINES protocols are unavailable.

## Options

#### **Named Pipes**

Named Pipes support is required on Microsoft Windows NT® and Microsoft Windows® 2000 installations of SQL Server. Server-side Named Pipes is not supported on Microsoft Windows 98. By default, SQL Server listens on the standard pipe for Named Pipes Net-Library connections.

#### **Named Pipes name**

Paths for the default and named instances differ:

- Default instance: \\.\pipe\sql\query
- Named instance: \\.\pipe\MSSQL\$instancename\sql\query

After SQL Server is installed, you can change the pipe name.

## **TCP/IP Sockets**

This Net-Library allows SQL Server to communicate by using standard Windows Sockets as the IPC method across the TCP/IP protocol. By default, all installations of Microsoft SQL Server 2000 on all operating systems use the TCP/IP Sockets Net-Library.

Note the following when using TCP/IP Sockets:

- SQL Server uses UDP port 1434 to establish connections from SQL Server 2000 clients. This socket number is also reserved for SQL Server by Internet Assigned Number Authority (IANA).
- Do not use dynamic ports and do not set a proxy server address, because the port you are listening on can change at each service startup.

## **Port Number**

If you set SQL Server to listen on TCP/IP, type the TCP/IP port number in the Port number box only if you want SQL Server to listen on a port address different from the default address. This is the port that SQL Server listens on when accepting connections from TCP/IP Sockets clients. The default number for a default instance is 1433, the official IANA socket number for SQL Server. The port for a named instance is dynamically assigned when the instance is first started, unless you set an alternate port during setup.

## **Remote Winsock proxy address**

If you set SQL Server to listen on a proxy server using Microsoft Proxy Server over TCP/IP Sockets, type the proxy server address in the Remote WinSock proxy address box when you set up the TCP/IP Sockets Net-Library.

## Multiprotocol

The Multiprotocol Net-Library uses the Windows NT remote procedure call (RPC) facility. In addition, the Multiprotocol Net-Library:

• Communicates over most IPC mechanisms supported by Windows NT. Only TCP/IP Sockets, NWLink IPX/SPX, and Named Pipes are

considered tested and supported.

- Allows the use of Windows Authentication over all protocols that RPC supports.
- Supports encryption for user password authentication as well as data.
- Offers performance comparable to native IPC Net-Libraries for most applications.

#### **Enable Multiprotocol encryption**

Use Multiprotocol encryption only for compatibility with existing systems. The Secure Sockets Layer (SSL) encryption that can be enabled using the Server Network Utility (after running Setup) is a more comprehensive encryption solution. Multiprotocol encryption is not supported on Windows 98 servers.

**Note** The Multiprotocol Net-Library is not supported with named instances.

#### NWLink IPX/SPX

This Net-Library allows SQL Server to communicate using the NWLink IPX/SPX protocol.

#### **Novell Bindery Service Name**

If you set up SQL Server to listen on NWLink IPX/SPX, the Setup program prompts you for the Novell Bindery service name in which to register SQL Server on the Novell network. The default service name is the computer name of the server computer. The Net-Library allows Novell SPX clients to connect to SQL Server.

The server NWLink IPX/SPX Net-Library is not available on Windows 98 and Windows 95.

#### **AppleTalk ADSP**

The server AppleTalk (ADSP) Net-Library allows Apple Macintosh® clients to connect to SQL Server using native AppleTalk (as opposed to TCP/IP

Sockets).

**Note** The AppleTalk Net-Library has not been enhanced for SQL Server 2000 and runs at a SQL Server 7.0 level of functionality. This Net-Library will not be supported in a future release of SQL Server 2000 and is not supported on named instances.

## **Apple Talk Service Object**

If you set up SQL Server to listen on AppleTalk, Setup prompts you for the AppleTalk service object name. The AppleTalk service object name is assigned by your system administrator. It is not necessary to enter an AppleTalk zone because the local zone is used when registering the service.

The AppleTalk Net-Library is not supported on Windows 98 and Windows 95.

## **Banyan VINES**

SQL Server supports Banyan VINES Sequenced Packet Protocol (SPP) as the IPC method across the Banyan VINES IP network protocol. Banyan VINES support for clients and servers running Windows NT is available for SQL Server on the Intel® platform only; it is not available on Windows 98 and Windows 95.

**Note** The Banyan VINES Net-Library has not been enhanced and runs at a SQL Server 7.0 level of functionality. This Net-Library will not be supported in a future release of SQL Server 2000 and is not supported on named instances.

## Street Talk Service name

If you set up SQL Server to listen on Banyan VINES, the Setup program prompts you for a StreetTalk service name. This has the form *servicename@group@org*, where *servicename* is the StreetTalk computerbased service name used by SQL Server, *group* is the group, and *org* is the organization. The computer-based service name used by SQL Server must first be created by using the MSERVICE program included with your Banyan VINES software. Also, to start SQL Server, you must be logged in with administrative permissions.

## Enable protocol encryption for all libraries

Select this check box to enable protocol encryption for all network libraries.

To use protocol encryption, you must have a certificate on the server. For information about obtaining a certificate, see the Microsoft Windows documentation. If you do not have a certificate, you can enable encryption after installing SQL Server using the Server Network Utility.

## **Default Net-Library Settings**

**Note** TCP/IP networking must be enabled before running SQL setup.

All Net-Libraries are installed by the Setup program. The table shows the default server and client Net-Library settings by operating system.

	Server Net-Library	Client Net-Library
Operating system	settings	settings
Windows 98	TCP/IP Sockets, Shared	TCP/IP Sockets
	Memory	
Windows 95	Not applicable	TCP/IP Sockets
Windows NT 4.0 (Server	TCP/IP Sockets, Shared	TCP/IP Sockets,
and Workstation)	Memory, Named Pipes	Named Pipes
Windows 2000 (all	TCP/IP Sockets, Shared	TCP/IP Sockets,
versions)	Memory, Named Pipes	Named Pipes

## See Also

**Configuring Client Net-Libraries** 

**Net-Libraries and Network Protocols** 

# **Services Accounts**

Use the **Services Accounts** screen in Setup to assign a logon account to each of the two Microsoft® SQL Server<sup>™</sup> services, SQL Server and SQL Server Agent. Either the local system or the domain user account is used, and you can use the same account for each service. The default setting is to use the same account for each service, and to automatically start each service. To use the default setting, enter your domain password and click **Next**.

You can also customize settings for each service. You can enter one logon account for both services, or specify an account for each. To later change options set on the **Services Accounts** screen, run the Services application in Windows Control Panel.

**IMPORTANT** To create or maintain a Microsoft SQL Server<sup>™</sup> 2000 failover cluster, you must be logged on to the computer with administrator privileges, that is, be a member of the **Administrators** local group of the computer or domain. For clustering this means that you must be an administrator of all nodes of the cluster.

When running SQL Server 2000 on Microsoft Windows NT 4.0, in addition to being logged on as an administrator, you must configure both SQL Server and SQL Server Agent to run as administrator accounts.

## **Options**

#### Use the same account for each service. Auto start SQL Server Service.

The default option: One account is used for both SQL Server and SQL Server Agent. These services start automatically when the operating system starts.

#### Customize the settings for each service.

Allows you to use different settings for the two services.

#### Services

Select a service for which you want to customize settings.

### SQL Server

Select this option to customize settings for the service, Microsoft SQL Server.

### **SQL Server Agent**

Select this option to customize settings for the service, Microsoft SQL Server Agent.

#### **Service Settings**

Select service settings as required.

### Use the Local System account

The local system account does not require a password, does not have network access rights in Windows NT 4.0, and may restrict your SQL Server installation from interacting with other servers.

Note In Windows 2000, the local system account does allow network access.

### Use a Domain User account

A domain user account uses Windows Authentication to set up and connect to SQL Server. By default, account information appears for the domain user account currently logged on to the computer.

#### Username

Accept or change the domain username.

#### Password

Enter the domain password.

#### Domain

Accept or change the domain name.

#### **Auto Start Service**

Select this option to automatically start a service when your operating system starts. This option is available only when customizing the settings for each service.

The SQL Server Agent service is dependent on the SQL Server service in that you can autostart the SQL Server Agent service only if you autostart the

SQL Server service as well.

**Note** When you click **Back** in the **Services Accounts** dialog box, the window you return to reverts to the default options. Options specified earlier are not retained.

## See Also

Setting Up Windows Services Accounts

**Changing Passwords and User Accounts** 

# **Authentication Mode**

Use this screen to choose the security (authentication) mode you want to use for this installation of Microsoft® SQL Server<sup>™</sup> 2000. If you select **Mixed Mode**, you are prompted to enter and confirm the system administrator password. After successful connection to SQL Server, the security mechanism is the same for both modes.

## **Options**

#### Windows Authentication Mode

When a user connects through a Microsoft Windows® user account, SQL Server validates the account name and password using information in the Windows operating system.

### Mixed Mode (Windows Authentication and SQL Server Authentication)

Allows users to connect using Windows Authentication or SQL Server Authentication. Users who connect through a Microsoft Windows user account can make use of trusted connections (connections validated by Windows) in either Windows Authentication Mode or Mixed Mode. SQL Server Authentication is provided for backward compatibility.

#### Add password for the sa login

Enter and confirm the system administrator password.

#### Blank Password (not recommended)

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

## See Also

Adding a SQL Server Login

Assigning an **sa** Password Authentication Modes Creating Security Accounts

# **Choose Licensing Mode**

Use this dialog box to set the licensing mode enabling your clients to access this instance of Microsoft® SQL Server<sup>™</sup>. SQL Server 2000 supports two client access licensing modes, one for each device and another for each processor.

A device in this context can be a workstation, terminal, or any other device running a SQL Server application connected to an instance of SQL Server.

A processor refers to a central processing unit (CPU) installed on a computer running an instance of SQL Server 2000. One computer may have multiple processors installed, requiring multiple processor licenses.

Once a licensing mode is set, you cannot change modes. You can add device or processor licenses after installing SQL Server, using the SQL Server 2000 Licensing Setup utility in Control Panel. (Do not confuse this licensing utility with the Windows Licensing utility, also found in Control Panel.)

For more information about licensing modes, see the Microsoft license agreement for SQL Server 2000.

## Options

### **Licensing Mode**

If accessing this dialog box from Control Panel, the mode chosen during setup is selected by default, along with the number of devices or processors you have previously selected.

#### Per Seat for

The Per Seat licensing mode requires a Client Access License for each device that will access SQL Server 2000 Server. Per Seat is often more economical for networks in which clients connect to more than one server.

In the edit box, select the number of devices to license.

#### **Processor License for**

With Processor licensing, a license is needed for each processor installed on the computer running SQL Server. The Processor License allows any number of devices to access the server, whether through an Intranet or over the Internet.

Using Processor licensing, SQL Server 2000 can take advantage of each installed processor, and support an unlimited number of client devices. A customer that provides access to SQL Server databases over the Internet, or that has a large number of users, will generally choose the Processor License.

In the edit box, select the number of processors to license.

### Continue

Click the **Continue** button to complete the installation process, or after modifying the number of devices or processors you want to license.

# **Installing a Remote Configuration**

Microsoft® SQL Server<sup>™</sup> 2000 can be installed on a remote computer, that is, a computer other than the one on which Setup is running. Before performing a remote installation:

- Ensure that the local and remote computers are running Microsoft Windows NT® or Windows® 2000.
- Ensure that the local and remote computers have an Intel®-compatible processor.
- Ensure that you are logged on to the local computer with a user account that has administrative privileges on the remote computer.

A remote setup is much like a normal installation, with two additional dialog boxes:

- The **Remote Setup Information** dialog box, which is also used when Setup is run on a computer that is part of a cluster. For more information, see <u>Remote Setup Information</u>.
- The **Select Computer** dialog box, which allows you to select a remote computer from the list of computers within the connected domains. The list may include computers not available for this installation, because permission must be granted before installing on a remote computer.

A computer network name may be entered instead of choosing from the list.

### See Also

Computer Name

# **Remote Setup Information**

Remote setup information is required to define security in two different setup situations:

- When you choose to install Microsoft® SQL Server<sup>™</sup> 2000 on a remote computer.
- When Setup is run on a computer that is part of a cluster, even if you are not creating or maintaining a failover cluster installation of SQL Server.

For a remote installation, SQL Server Setup collects the information you enter in Setup dialog boxes, recording the entries into the Setup.iss file. At the same time, the remote setup process starts a remote service, copies files to the \admin\$ share directory, and runs an unattended installation on the remote computer using the options specified in Setup.iss.

For clustered computers, the **Remote Setup Information** box is displayed because any installation on a failover cluster system needs the administrator account to install Microsoft Distributed Transaction Coordinator on both nodes, or to verify the presence of MS DTC. Administrator information must be entered that is valid for all selected nodes in the failover cluster system.

## **Options**

#### Username, Password, and Domain

Specify the user account under which SQL Server Setup starts a service on the remote computer. This user account must be an administrator on the remote computer and have read access to the Setup source files directory.

Do not confuse the user account entered on this screen with:

- The user account logged on to the local computer.
- The user account assigned in SQL Server Setup to the SQL Server and SQL Server Agent services.

Each of these user accounts is specified separately. However, you can use the same user information in each case. That is, you can use the same name, password, and Windows domain for each account.

#### **Target computer**

The name of the remote computer entered in the **Computer Name** dialog box is shown in static text.

#### **Target path**

The name of the remote computer and, in Universal Naming Convention format, the directory on the remote computer where SQL Server is to be installed. For example:

\\target\_computer\C\$\Program Files\Microsoft SQL Server

#### **Setup Source Files**

Location of the setup program files used for the remote installation.

#### To perform a remote installation

# **Advanced Installation Options**

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

## **Options**

### **Record Unattended .ISS file**

Create a setup initialization file for unattended installations.

### **Registry Rebuild**

Rebuild registry for a corrupted installation.

### Maintain a virtual server for failover clustering

Make changes to existing clusters, such as revising the name, or adding and removing cluster nodes.

## See Also

Performing an Unattended Installation

**Rebuilding the Registry** 

Installing a Virtual Server Configuration

SQL Server Language Support

# **Installing a Virtual Server Configuration**

The topics in this section provide information about the Setup screens used in setting up and maintaining failover clustering.

# **Failover Clustering: Defining the Virtual Server**

Use the **Failover Clustering** screen to define the virtual server for a new cluster, or to maintain the virtual server definition for an existing cluster. You can add and remove IP addresses; multiple IP addresses are allowed for each virtual server.

## **Options**

#### Virtual Server Name

Displays the network name of the virtual server. This is the name users will see when they connect to the virtual server.

When upgrading to a cluster, this name is entered in the **Virtual Server Name** dialog box.

#### **IP address**

Enter the IP address or addresses used to connect to the virtual server.

#### SubNet

Displays the Subnet, which is supplied by MSCS.

#### Network

Displays the Network name you assigned each subnet during setup of MSCS.

### Add

Adds the specified IP address and SubNet to the named virtual server.

#### Remove

Removes the specified IP address and SubNet from the named virtual server.

## See Also

Before Installing Failover Clustering

Creating a Failover Cluster

Upgrading to a SQL Server 2000 Failover Cluster

# **Cluster Management Screen**

Use the **Cluster Management** screen to review the cluster definition provided by Microsoft® SQL Server<sup>™</sup> 2000, and make changes if necessary. After you have specified nodes for the virtual server, Setup installs or uninstalls the SQL Server binary files on each node.

**CAUTION** If you modify the node list of a virtual server using the quorum resource, your cluster may not fail over properly. For more information, see <u>Modify Node List Warning</u>.

## **Options**

### **Available Nodes**

A list of computers that can be added to the current virtual server definition. If a computer you want is not available at this time, you can run Setup later to add it to the virtual server definition.

#### **Configured Nodes**

List of computers currently configured in the current virtual server definition. The computer at the top of the list is the preferred node.

#### **Unavailable Nodes**

Computers that are currently offline or not available to be added to a cluster definition.

### Add

Adds the selected available node to the list of configured nodes.

#### Remove

Removes the selected configured node from the list of configured nodes.

## See Also

**Failover Clustering** 

Maintaining a Failover Cluster

Cluster Disk Selection Screen

Upgrading to a SQL Server 2000 Failover Cluster

# **Cluster Disk Selection Screen**

Use the **Cluster Disk Selection** screen to select a cluster group during the installation of a new virtual server or during an upgrade to a cluster. A cluster group is composed of one or more shared cluster disks within a group, and can contain at most one Microsoft® SQL Server<sup>TM</sup> virtual server. The **Cluster Disk Selection** screen lists only those groups that already have the shared cluster disk added as a resource. For more information about cluster disks, see <u>Creating a Failover Cluster</u>.

**CAUTION** Do not select the quorum disk (the last group in the list) because the quorum disk must be treated as a special resource. Clustering may fail if selected as a cluster group. A warning message appears if you select the quorum disk. For more information, see <u>Quorum Disk Selection Warning</u>.

When using a small cluster, the quorum disk may be the only choice available. Use it only for testing purposes or to explore failover clustering.

**IMPORTANT** Never use the quorum group for production purposes.

### See Also

**Failover** Clustering

Maintaining a Failover Cluster

Modify Node List Warning

# **Quorum Disk Selection Warning**

The following warning message appears if you select the quorum disk (the last group in the list) on the **Cluster Disk Selection** screen. This warning applies to both Microsoft® Windows NT® 4.0 and Microsoft Windows® 2000:

It is strongly recommended that you not use the quorum group with SQL Server.

The quorum disk is a special resource in the Windows operating system. If you select the quorum disk, you may later want to restrict ownership of Microsoft® SQL Server<sup>TM</sup> to a subset of the cluster nodes. However, the quorum group owner list must include all of the nodes in the cluster.

For example, you may have a two-node cluster (Node1 and Node2) with SQL Server set to use the quorum disk group. If you then modify SQL Server to have only Node1 in the virtual server definition, the quorum disk group is prevented from failing over to Node2. In the event of a failure of Node1, the result is that you not only lose the virtual SQL Server, but the entire MSCS cluster.

This is true for Windows NT 4.0 only. In Windows 2000 the node list is ignored and the quorum disk group can fail over to any node in the cluster configuration. However, another issue may arise. In the previous example, there are no SQL Server program files available on Node2, but the cluster group can fail over to Node2. In this situation, SQL Server is unable to run on Node2 but the cluster group containing the quorum disk may fail over anyway, making your SQL Server unavailable.

For more information about the quorum disk, see the Windows NT documentation.

## See Also

<u>Cluster Disk Selection Screen</u> <u>Modify Node List Warning</u> <u>Creating a Failover Cluster</u> Failover Clustering Dependencies

# **Modify Node List Warning**

In the **Cluster Management** screen, use caution if you modify the node list of a Microsoft® SQL Server<sup>™</sup> 2000 virtual server using the quorum resource. If such a node list is modified, the following warning appears when **Next** is clicked:

Modifying the node list of the quorum resource may prevent your cluster from failing over properly. Are you sure you want to do this?

The quorum resource itself is unable to fail over to any servers that you did not select as part of your virtual server definition. This may jeopardize the availability of your failover cluster. For more information, see the Microsoft Windows NT® documentation.

**Note** This problem does not occur when you run SQL Server 2000 on Microsoft Windows® 2000.

## See Also

Quorum Disk Selection Warning Creating a Failover Cluster Failover Clustering Dependencies

# **Performing an Unattended Installation**

You can perform an unattended installation of Microsoft® SQL Server<sup>™</sup> 2000, in which setup screen entries are made automatically using stored information. An unattended installation can be convenient if you want to perform several installations of SQL Server with identical configurations on different computers. An unattended installation requires a setup initialization file, which can be created in several different ways.

By default, each time you install SQL Server using the Setup screens, the options you select are recorded into the setup initialization file, Setup.iss. Setup.iss is placed in the system root directory (%windir%), and is available to provide installation settings at a later time.

**Note** You cannot perform an unattended installation to set up a failover cluster of Microsoft SQL Server 2000.

## **Creating a Setup File Using the Record Unattended Option**

In Setup, when you select the **Record Unattended .ISS file** option in the **Advanced Options** screen, each subsequent choice you make in the setup screens is recorded in the Setup.iss file stored in the system root directory. SQL Server files are not installed in this process. The Setup.iss file can then be run as is, or revised in a text editor if necessary.

#### To record an unattended installation file

# **Creating a Setup File Manually**

You create a customized setup initialization file interactively when you select the **Record Unattended** option in Microsoft® SQL Server<sup>™</sup> 2000 Setup. You can also edit files manually, to further refine and customize setup initialization files.

## Creating or Modifying a Setup File Using a Text Editor

You can use a text editor to modify the Setup.iss file generated using the **Record Unattended .ISS file** option. You can also modify one of the sample setup files (\*.iss) included on the SQL Server 2000 compact disc or you can create your own setup file.

To modify one of the sample setup initialization files found on the SQL Server compact disc, open the file in a text editor and modify as required. Keep the file compatible with the Microsoft Windows® initialization file format and save it with the .iss file name extension.

# Format of a Sample Setup Initialization File

A setup initialization file is a text file that uses the standard Windows .ini file format. Sections of the sample setup initialization file for a typical installation of Microsoft SQL Server 2000 are described in the tables that follow. This sample file (Sqlins.iss) is found in the root directory of the SQL Server compact disc.

**Note** In creating a setup file for a named instance of SQL Server 2000, you must indicate the instance name you want to install and the path required to navigate through the setup screens. The **Instance Name** dialog box [DlgInstanceName] section must be modified, as well as other places in the setup file where the instance name appears.

# [InstallShield Silent]

This section is required for InstallShield. Do not change the values.

Entry	Value	Description
Version	v5.00.000	Version of the InstallShield

		Silent response file.
File	Response File	Indicates this is the
		Response File.

## [File Transfer]

Entry	Value	Description
OverwriteReadOnly	NoToAll	Do not overwrite read-only
		files.

# [DlgOrder]

Lists each dialog box in the order it appears in an attended setup. The listing in this section must correspond to the other sections in the setup initialization file.

Entry	Value	Description
Dlg0	SdWelcome-0	Initial dialog box
Count	14	Number of dialog boxes
		listed in this section
Dlg1	DlgMachine-0	Next dialog box
Dlg2	DlgInstallMode-0	Next dialog box
Dlg3	SdRegisterUser-0	Next dialog box
Dlg4	SdLicense-0	Next dialog box
Dlg5	CDKEYDialog-0	Next dialog box
Dlg 6	DlgClientServer-0	Next dialog box
Dlg7	DlgInstanceName-0	Next dialog box
Dlg8	SetupTypeSQL-0	Next dialog box
Dlg9	DlgServices-0	Next dialog box
Dlg10	DLGSqlSecurity-0	Next dialog box
Dlg11	DlgCollation-0	Next dialog box
Dlg12	DlgServerNetwork-0	Next dialog box
Dlg13	SdStartCopy-0	Next dialog box
Dlg14	SdFinish-0	Last dialog box

## [SdWelcome-0]

Corresponds to the **Welcome** dialog box.

Entry	Value	Description
Result	1	Next

## [DlgMachine-0]

Corresponds to the **Computer Name** dialog box.

Entry	Value	Description
Туре	1	Local computer
Result	1	Next

# [DlgInstallMode-0]

Corresponds to the **Installation Selection** dialog box.

Entry	Value	Description
Туре	1	Create a new instance
Result	1	Next

## [SdRegisterUser-0]

Corresponds to the **User Information** dialog box.

Entry	Value	Description
szName	<user name=""></user>	Name of user; company
		name is not required.
Result	1	Next.

## [SdLicense-0]

Corresponds to the **Software License Agreement** dialog box.

Entry	Value	Description
Result	1	Yes

## [CDKEYDialog-0]

Corresponds to the **CD-Key** dialog box.

Entry	Value	Description
svCDKey	<cd key="" value=""></cd>	Specified for each
		installation
Result	1	Next

# [DlgClientServer-0]

Corresponds to the **Installation Definition** dialog box.

Entry	Value	Description
Туре	2	Server and client tools
Result	1	Next

## [DlgInstanceName-0]

Corresponds to the **Instance Name** dialog box.

Entry	Value	Description
InstanceName	MSSQLSERVER	Designation of the default
		instance (always the same).

InstanceName	<instance name=""></instance>	Designation of a named
		instance.
Result	1	Next.

# [SetupTypeSQL-0]

Corresponds to the **Setup type** dialog box.

Entry	Value	Description
szDir	%PROGRAMFILES%\Microsoft	Directory where SQL
	SQL Server	Server program files are
		installed.
Result	301	Typical (301)
		(302 = Minimum and
		303 = Custom).
szDataDir	%PROGRAMFILES%\Microsoft	Directory where SQL
	SQL Server	Server data files are
		installed (same as
		program files).

## [DlgServices-0]

Corresponds to the **Services Accounts** dialog box.

Entry	Value	Description
Local-Domain	3855	Use the same account for
	<other numeric="" value=""></other>	each service.
		To customize the settings
		for each service, see <u>Setup</u>
		Initialization File Details.
AutoStart	15	Autostart Service is
		enabled.
Result	1	Next.

## [DlgSQLSecurity-0]

Corresponds to the **Authentication** dialog box. Choices shown here include options not in the sample Sqlins.iss file.

Entry	Value	Description
LoginMode	-1	System default security is used.
		The Microsoft Windows
		NT <sup>®</sup> default is Windows
		Authentication Mode.
		The Microsoft Windows 98
		default is Mixed Mode,
		with no <b>sa</b> password.
LoginMode	1	Windows Authentication
		Mode.
LoginMode	2	Mixed Mode.
szPwd	<choice of="" password=""></choice>	Used only with Mixed
		Mode security.
Result	1	Next.

## [DlgCollation-0]

Corresponds to the **Collation Settings** dialog box.

Entry	Value	Description
collation_name	1 1	When blank, system default
		collation is used.
collation_name	<collation designator=""></collation>	Selected by user. For more
		information, see <u>Collation</u>
		<u>Settings in Setup</u> .

#### [SdServerNetwork-0]

Entry	Value	Description
NetworkLibs	255	Named pipes and TCP/IP
	245	Value 245 = TCP/IP only
	15	Value 15 = Named pipes
		only
		To customize network
		library settings, see <u>Setup</u>
		Initialization File Details
TCPPort	1433	Port address, for TCP/IP
TCPPrxy	Default	Default proxy, or what is
		entered
NMPPipeName	\\.\pipe\sql\query	Pipe name
Result	1	Next

Corresponds to the **Network Libraries** dialog box.

## [SdStartCopy-0]

Corresponds to the **Start Copying Files** dialog box.

Entry	Value	Description
Result	1	Next

#### [SdFinish-0]

Corresponds to the **Setup Complete** dialog box.

Entry	Value	Description
Result	1	Next
bOpt1	0	Placeholder for stock dialog
		box
bOpt2	0	Placeholder for stock dialog

	b	OX

# See Also

Performing an Unattended Installation

Setup Initialization File Details

# **Setup Initialization File Details**

When creating a customized setup initialization file, the **Service Accounts** and **Network Libraries** dialog boxes have additional options used for an unattended installation.

## **Services Accounts Dialog Box**

To customize settings for each service, you can calculate values for the Local-Domain and AutoStart entries.

## Local-Domain

If you want the SQL Server and SQL Server Agent services to use different logon accounts, you can calculate the value to enter for Local-Domain. The Local-Domain value is a bitwise logical OR combination of the values shown in the following table. For more information, see <u>(Bitwise OR)</u>.

Service	Account to use	Hexadecimal value
SQL Server	Local System account	x000000F
SQL Server Agent	Local System account	0x00000F00
SQL Server	Domain User account	0x00000F0
SQL Server Agent	Domain User account	0x0000F000

Additional entries must be added to your setup initialization file if the **Domain User** account is used for either service.

When SQL Server service is using a domain account, Setup looks for values for:

- SQLDomain = <domain name>
- SQLDomainAcct = <domain user account>
- SQLDomainPwd = <domain password: an encrypted password available only using setup screens>

When SQL Server Agent service is using a domain account, Setup looks for values for:

- AgtDomain = <domain name>
- AgtDomainAcct = <domain user account>
- AgtDomainPwd = <domain password: an encrypted password available only using setup screens>

### AutoStart

The value for the **AutoStart** option is a bitwise logical OR combination using the following hexadecimal values:

Autostart SQL Server = 0x000000F

Autostart SQL Server Agent = 0x000000F0

A value of zero (0) for either service indicates no AutoStart; manual startup is required.

### **Network Libraries Dialog Box**

To customize network library settings, you can calculate a value for the NetworkLibs entry in the **Network Libraries** dialog box. The value for NetworkLibs is a bitwise logical OR combination of the values shown in the following table. When a network library is set, additional information must be entered in the setup initialization file, as shown in the third column.

Network Library to use	Hexadecimal value	
when connecting to the	to use in Bitwise	Additional information
server	OR operation	that Setup looks for
Named Pipes	0xF	NMPPipeName = <named< td=""></named<>
		pipe name>
TCP/IP Sockets	0xF0	TCPPort = <port number=""></port>
		TCPPrxy = <remote< td=""></remote<>

		Winsock proxy address>
NW Link	0xF000	NWLinkObj = <novell< td=""></novell<>
		Bindary service name>
Apple Talk	0xF0000	ApplObj = <apple talk<br="">service object&gt;</apple>
Banyan VINES	0xF00000	BanyanObj = <streettalk service name&gt;</streettalk 

## See Also

Services Accounts

Network Libraries

Performing an Unattended Installation

# **Installing SQL Server Using SMS**

You can use Microsoft® Systems Management Server (SMS) version 1.2 or later to install Microsoft SQL Server<sup>™</sup> 2000 automatically on multiple server computers running Microsoft Windows NT® or Microsoft Windows® 2000 in your enterprise.

The SQL Server compact disc contains a Package Definition Format (PDF) file (Smssql.pdf) that automates creating a SQL Server package in SMS. The SQL Server package can then be distributed and installed on SMS computers.

Smssql.pdf includes instructions for running the batch file Smssqins.bat with Sqlins.iss (the setup initialization file) for a typical installation. Both of these files are included on the SQL Server compact disc.

To create a custom command file, edit a copy of Smssql.pdf.

#### See Also

Performing an Unattended Installation

Creating a Setup File Manually

# **Rebuilding the Registry**

The **Registry Rebuild** option on the **Advanced Options** Setup screen allows you to rebuild the registry for a corrupted Microsoft® SQL Server<sup>™</sup> installation. This process fixes only the registry; it does not fix data errors or the **master** database.

**IMPORTANT** To rebuild the registry, you must enter setup information using the same choices that you entered during the initial installation. If you do not know or are not sure of this information, do not use this registry rebuild process. To restore the registry, you must uninstall and reinstall SQL Server.

#### To rebuild the registry

# Working with Named and Multiple Instances of SQL Server 2000

With Microsoft® SQL Server<sup>™</sup> 2000, you have the option of installing multiple copies, or instances of SQL Server on one computer. When setting up a new installation of SQL Server 2000 or maintaining an existing installation, you can specify it as:

• A default instance of SQL Server.

This instance is identified by the network name of the computer on which it is running. Applications using client software from earlier versions of SQL Server can connect to a default instance. SQL Server version 6.5 or SQL Server version 7.0 servers can operate as default instances. However, a computer can have only one version functioning as the default instance at a time.

• A named instance of SQL Server.

This instance is identified by the network name of the computer plus an instance name, in the format *<computername>\<instancename>*. Applications must use SQL Server 2000 client components to connect to a named instance. A computer can run any number of named instances of SQL Server concurrently. A named instance can run at the same time as an existing installation of SQL Server version 6.5 or SQL Server version 7.0. The instance name cannot exceed 16 characters.

A new instance name must begin with a letter, an ampersand (&), or an underscore (\_), and can contain numbers, letters, or other characters. SQL Server **sysnames** and reserved names should not be used as instance names. For example, the term "default" should not be used as an instance name because it is a reserved name used by Setup.

Single and multiple instances of SQL Server 2000 (default or named) are available using the SQL Server 2000 Personal Edition, the SQL Server 2000 Standard Edition, or the SQL Server 2000 Enterprise Edition.

## **Default Instances**

You cannot install a default instance of SQL Server 2000 on a computer that is also running SQL Server 7.0. You must either upgrade the SQL Server 7.0 installation to a default instance of SQL Server 2000, or keep the default instance of SQL Server 7.0 and install a named instance of SQL Server 2000.

You can install a default instance of SQL Server 2000 on a computer running SQL Server 6.5, but the SQL Server 6.5 installation and the default instance of SQL Server 2000 cannot be running at the same time. You must switch between the two using the SQL Server 2000 **vswitch** command prompt utility.

# **Multiple Instances**

Multiple instances occur when you have more than one instance of SQL Server 2000 installed on one computer. Each instance operates independently from any other instance on the same computer, and applications can connect to any of the instances. The number of instances that can run on a single computer depends on resources available. The maximum number of instances supported in SQL Server 2000 is 16.

When you install SQL Server 2000 on a computer with no existing installations of SQL Server, Setup specifies the installation of a default instance. However, you can choose to install SQL Server 2000 as a named instance instead by clearing the **Default** option in the **Instance Name** dialog box.

A named instance of SQL Server 2000 can be installed at any time: before installing the default instance of SQL Server 2000, after installing the default instance of SQL Server 2000, or instead of installing the default instance of SQL Server 2000.

Each named instance is made up of a distinct set of services and can have completely different settings for collations and other options. The directory structure, registry structure, and service names all reflect the specific instance name you specify.

## See Also

Multiple Instances of SQL Server

Naming Conventions for Instances of SQL Server 2000 Network Protocols for Named Instances File Locations for Multiple Instances of SQL Server Working with Instances and Versions of SQL Server

# Naming Conventions for Instances of SQL Server 2000

Because Microsoft<sup>®</sup> SQL Server<sup>™</sup> 2000 can be set up to include one or more named instances, with or instead of a default instance, new naming conventions are used to distinguish between instances.

In earlier versions, a SQL Server installation is identified by computer name. In SQL Server 2000, only the default instance is identified solely by computer name. A named instance is identified by a combination of computer name and instance name. This instance name is also reflected in the names of the associated SQL Server services.

**Note** There can be only one default instance of SQL Server for each computer. It can be an intact SQL Server version 6.5 or SQL Server version 7.0 installation, or it can be an installation of SQL Server 2000 set up as the default instance. In either case, the default instance uses the same service names, registry structure, network listening points, and other defaults used in SQL Server 7.0.

## Service Names for Default and Named Instances

When you install a default instance of SQL Server, the service names remain MSSQLServer and SQLServerAgent (the same as in SQL Server 7.0).

When you install a named instance of SQL Server, the service names are changed to:

- MSSQL\$InstanceName for the MSSQLServer service.
- SQLAgent\$InstanceName for the SQLServerAgent service.

The Microsoft Distributed Transaction Coordinator and Microsoft Search services are installed only once, and can be used simultaneously by every installed instance of SQL Server.

## See Also

Multiple Instances of SQL Server

# **Network Protocols for Named Instances**

When you install a default instance of Microsoft® SQL Server<sup>™</sup> 2000, the standard network addresses are enabled. For example, named pipes uses \\.\pipe\sql\query, and TCP/IP sockets connect to port 1433.

When you select a named instance, only the Named Pipes, TCP/IP, and NWLink IPX/SPX protocols are supported. Named Pipes defaults to a network address of \\Computername\Pipe\MSSQL\$instancename\Sql\Query. The port addresses used by TCP/IP and NWLink IPX/SPX are chosen dynamically (by default) the first time the instance is started.

## See Also

Communicating with Multiple Instances Multiple Instances of SQL Server Network Libraries

# **File Locations for Multiple Instances of SQL Server**

Each named instance of Microsoft® SQL Server<sup>™</sup> 2000 has a specific location for its program files and another for its data files that is different from that of the default instance of SQL Server.

**Note** A named instance is not necessarily the same as a multiple instance. You can have a single named instance or you can have multiple named instances. For more information, see <u>Multiple Instances of SQL Server</u>.

For each named instance of SQL Server that you install, the default directories are:

- \Program Files\Microsoft SQL Server\MSSQL\$InstanceName\Binn for executable files.
- \Program Files\Microsoft SQL Server\MSSQL\$InstanceName\Data for data files.

Shared tools for all instances, both default and named instances, are located in the \Program Files\Microsoft SQL Server\80\Tools directory. You can specify file paths other than the default locations for program and data file for multiple instances.

The following illustration shows the simplest case of multiple instances of Microsoft SQL Server 2000: the default instance and one named instance, **Instance1**. A named instance has its own full set of data files and executable files. Common files used by both the default instance and any named instances are installed in the folder \Program Files\Microsoft SQL Server\80.

**Note** If Microsoft SQL Server version 7.0 is used as the default installation alongside a named instance of SQL Server 2000, program and data files are located at C:\Mssql7, the default location for SQL Server 7.0 files.

### **Finding Install Locations**

If you are uncertain about instance paths, query the registry to get the installation

path of a particular instance. Run the following at the command prompt, inserting the appropriate instance name:

C:\> REG QUERY HKLM\Software\Microsoft\Microsoft SQL Server\

**Note** The REG QUERY tool is available in the Microsoft Windows® 2000 Resource Kit.

#### See Also

<u>File Paths for SQL Server 2000</u> <u>Multiple Instances of SQL Server</u>

# **Removing Multiple Instances of SQL Server 2000**

When you remove a default or named instance of Microsoft® SQL Server<sup>TM</sup> 2000, the data files and registry keys for that instance are deleted. Tools cannot be removed until all instances of SQL Server 2000 have been removed from a computer, because the tools are shared among all installed instances.

To remove a single instance of SQL Server 2000, or to remove all installed instances, see <u>How to remove SQL Server 2000 (Windows)</u>.

# **Working with Instances and Versions of SQL Server**

Multiple instances in Microsoft® SQL Server<sup>™</sup> 2000 offer enhanced ways to work with earlier versions of Microsoft SQL Server already installed on your computer. You can leave previous installations intact, and also install and run SQL Server 2000. For example, you can run SQL Server version 7.0 and a named instance of SQL Server 2000 at the same time, or you can run SQL Server version 6.5 in a version switch configuration with SQL Server 2000. If you need to have three different versions of SQL Server installed on the same computer, there are several ways to accomplish this.

In addition, users of all editions of SQL Server can have more than one instance of SQL Server 2000 installed and running at once (multiple instances), as well as one or more earlier versions.

Considerations for using SQL Server 2000 in combination with previous installations include:

- Using SQL Server 6.5 with the default instance or named instances of SQL Server 2000.
- Running SQL Server 7.0 with a named instance of SQL Server 2000.
- Working with three versions of SQL Server: SQL Server 6.5, SQL Server 7.0, and SQL Server 2000.

**Note** The concept of the default instance is new to SQL Server 2000, due to the introduction of multiple instances. If installed on the same computer as SQL Server 2000, either SQL Server version 6.5 or SQL Server version 7.0 can function as default instances of SQL Server. (A default instance is identified by the network name of the computer on which it is running.) For more information, see <u>Working with Named and Multiple Instances of SQL Server</u> 2000.

### **Using SQL Server Books Online for SQL Server 7.0**

When you keep Microsoft SQL Server version 7.0 on your computer and install a named instance of SQL Server 2000, SQL Server Books Online for SQL Server 7.0 remains in its original location: C:\Mssql7\Books. In this side-by-side configuration, Books Online for SQL Server 7.0 remains accessible from the start menu in the SQL Server 7.0 program group.

**Note** This is an exception to what occurs for the other shared tools (such as code samples, scripts, and templates), when a named instance of SQL Server 2000 is installed along with SQL Server 7.0. All other shared tools from the 7.0 installation are copied to storage locations, with pointers to the SQL Server 2000 tools replacing previous versions of the tools. Files for Books Online for SQL Server 7.0 are not redirected in this way -- they remain ready for use.

When SQL Server 7.0 is upgraded to the default version of SQL Server 2000, the 7.0 Books Online files are also upgraded. That is, they are replaced with the SQL Server 2000 Books Online.

Whether you have SQL Server 7.0 installed or not, you can access information in the SQL Server 7.0 documentation. For more information, see <u>How to access</u> <u>SQL Server Books Online for SQL Server 7.0</u>.

#### See Also

Using SQL Server 6.5 with SQL Server 2000

Running SQL Server 7.0 Along with a Named Instance of SQL Server 2000

Working with Three Versions of SQL Server

# Using SQL Server 6.5 with SQL Server 2000

If you have Microsoft® SQL Server<sup>™</sup> version 6.5 installed, you can keep the SQL Server version 6.5 installation and also install a default or named instance of SQL Server 2000. No version upgrading is involved; however, version switching can be used to move between SQL Server version 6.5 and SQL Server 2000. In addition, SQL Server 2000 tools are used to control both SQL Server 2000 and SQL Server version 6.5.

**WARNING** After SQL Server 2000 is installed, the SQL Server version 6.5 Trace utility and other earlier tools are no longer available.

To install SQL Server 2000 alongside SQL Server 6.5:

- Keep your SQL Server 6.5 configuration intact.
- Install SQL Server 2000, selecting either a default or named instance in the **Instance Name** dialog box.
- Switch versions from SQL Server 6.5 to the default instance of SQL Server 2000.

**Note** Switching from SQL Server 2000 back to SQL Server 6.5 is not recommended.

The illustration shows an installation of SQL Server 6.5 in a version switch configuration with SQL Server 2000.

	Default instance of SQL Server 2000
Version switch	
	SQL Server version 6.5

## See Also

Switching Between SQL Server 6.5 and SQL Server 2000 Running SQL Server 7.0 Along with a Named Instance of SQL Server 2000 Working with Three Versions of SQL Server

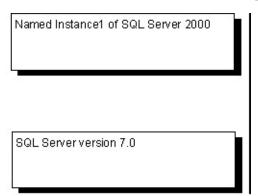
# **Running SQL Server 7.0 Along with a Named Instance of SQL Server 2000**

You can keep an installation of Microsoft® SQL Server<sup>™</sup> version 7.0 intact on your computer and also install a named instance of SQL Server 2000 on the same computer. This configuration enables you to run both the original installation of SQL Server 7.0 and the named instance of SQL Server 2000 at the same time, without using the **vswitch** command prompt utility.

To run a named instance of SQL Server 2000 with an existing SQL Server 7.0 installation intact:

- Keep SQL Server version 7.0 in its original condition with no version upgrade to SQL Server 2000. SQL Server 7.0 functions as the default instance of SQL Server, identified by the network name of the computer.
- Install a named instance of SQL Server 2000, identified by both the network name of the computer plus an instance name.

The illustration shows this configuration.



#### See Also

How to install a named instance of SQL Server 2000 (Setup) Working with Three Versions of SQL Server

## **Working with Three Versions of SQL Server**

This topic describes two scenarios for working with SQL Server version 6.5, SQL Server version 7.0, and SQL Server 2000. One example shows three versions installed at one time, with no version upgrades, but with a version switch between SQL Server 6.5 and SQL Server 7.0. The other involves upgrading to SQL Server 2000 from SQL Server 7.0, and then version switching between SQL Server 6.5 and SQL Server 2000.

In any of these situations, multiple named instances of SQL Server 2000 can be installed as well. However, only two different versions of SQL Server can run at one time, using version switching in one of two ways:

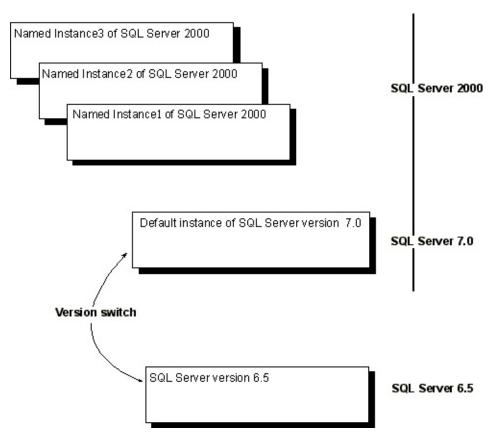
- Switch between SQL Server 6.5 and SQL Server 7.0.
- Switch between SQL Server 6.5 and SQL Server 2000.

#### **Using Version Switching**

To use version switching with SQL Server 6.5 and SQL Server 7.0, while at the same time running multiple instances of SQL Server 2000:

- Keep the SQL Server 6.5 configuration intact.
- Keep the SQL Server 7.0 configuration intact, with no version upgrade to SQL Server 2000.
- Install one or more named instances of SQL Server 2000.

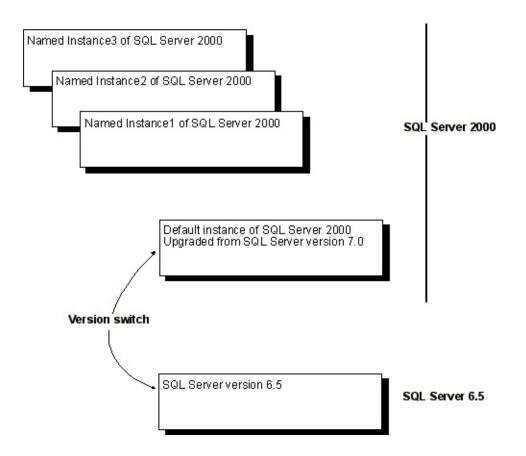
The illustration shows how named instances of SQL Server 2000 and the existing installation of SQL Server version 7.0 can run at the same time. SQL Server version 6.5 is available to be switched in as the default instance instead of SQL Server 7.0.



To use version switching with SQL Server 6.5 and SQL Server 2000, after upgrading from SQL Server version 7.0:

- Keep the SQL Server 6.5 configuration intact.
- Have SQL Server 7.0 installed, but prepare to upgrade SQL Server 7.0 to SQL Server 2000.
- Run Setup. When SQL Server 7.0 is detected, upgrade SQL Server 7.0 to the default instance of SQL Server 2000. (Select the option to Upgrade in the **Existing Installation Options** dialog box, and leave the Default check box selected in the **Instance Name** dialog box.) At this point, the installation of SQL Server 7.0 no longer exists; it is replaced by the default instance of SQL Server 2000.

The illustration shows this configuration, along with three SQL Server 2000 named instances.



#### See Also

Upgrading from SQL Server 7.0 to SQL Server 2000

Using SQL Server 6.5 with SQL Server 2000

Running SQL Server 7.0 Along with a Named Instance of SQL Server 2000

# **Failover Clustering**

In Microsoft® SQL Server<sup>™</sup> 2000 Enterprise Edition, SQL Server 2000 failover clustering provides high availability support. For example, during an operating system failure or a planned upgrade, you can configure one failover cluster to fail over to any other node in the failover cluster configuration. In this way, you minimize system downtime, thus providing high server availability.

To install, configure, and maintain a failover cluster, use SQL Server Setup. For information about upgrading to a SQL Server 2000 failover cluster, see <u>Upgrading to a SQL Server 2000 Failover Cluster</u>.

Use failover clustering to:

• Install SQL Server on multiple nodes in a failover cluster. You are limited only by the number of nodes supported by the operating system.

Before installing failover clustering, you must install Microsoft Windows NT® 4.0, Enterprise Edition, Microsoft Windows® 2000 Advanced Server or Windows 2000 Datacenter Server, and the Microsoft Cluster Service (MSCS).

There are specific installation steps that must be followed to use failover clustering. For more information, see <u>Installing Failover Clustering</u> and <u>Handling a Failover Cluster Installation</u>.

• Specify multiple IP addresses for each virtual server.

SQL Server 2000 allows you to use all available network IP subnets, thereby providing alternate ways to connect if one subnet fails and increasing network scalability. For example, with a single network adaptor, a network failure can disrupt communications. However, with multiple network cards in the server, each network can be on a different IP subnet. If one subnet fails, at least one connection can continue to function. If a router fails, MSCS continues to function, and all IP addresses still work. However, if the network card on the local computer fails, communication still may be disrupted. For more information, see <u>Creating a Failover Cluster</u>.

• Administer a failover cluster from any node in the clustered SQL Server

configuration. To perform setup tasks, you must be working from the node in control of the cluster disk resource. For more information, see <u>Creating a Failover Cluster</u>.

- Allow one virtual server to fail over to any other node on the failover cluster configuration. For more information, see <u>Creating a Failover</u> <u>Cluster</u>.
- Add or remove nodes from the failover cluster configuration using the Setup program. For more information, see <u>Maintaining a Failover</u> <u>Cluster</u>.
- Reinstall or rebuild a virtual server on any node in the failover cluster without affecting the other nodes. For more information, see <u>Maintaining a Failover Cluster</u>.
- Perform full-text queries by using Microsoft Search service with failover clustering. For more information, see <u>Using SQL Server Tools</u> with Failover Clustering.

## **Multiple Instance Support**

Failover clustering also supports multiple instances. Multiple instance support makes it easier to build, install, and configure virtual servers in a failover cluster. Applications can connect to each instance on a single computer in much the same way as they connect to instances of SQL Server running on multiple computers. For more information about virtual servers, see <u>Creating a Failover Cluster</u>.

With multiple instance support, you can isolate work environments (for example, testing from production) or volatile application environments and provide different system administrators for each instance of SQL Server on the same computer. For more information, see <u>Multiple Instances of SQL Server</u>.

#### See Also

Failover Clustering Architecture

# **Failover Clustering Support**

In Microsoft® SQL Server<sup>™</sup> 2000 Enterprise Edition, the number of nodes supported in SQL Server 2000 failover clustering depends on the operating system you are running:

- Microsoft Windows NT® 4.0, Enterprise Edition, Microsoft Windows® 2000 Advanced Server, and Microsoft Windows 2000 Datacenter Server support two-node failover clustering.
- Windows 2000 Datacenter Server supports up to four-node failover clustering, including an active/active/active/active failover clustering configuration.

The following tools, features and components are supported with failover clustering:

- Microsoft Search service. For more information, see <u>Using SQL Server</u> <u>Tools with Failover Clustering</u>.
- Multiple instances. For more information, see <u>Failover Clustering</u>.
- SQL Server Enterprise Manager. For more information, see <u>Using SQL</u> <u>Server Tools with Failover Clustering</u>.
- Service Control Manager. For more information, see <u>Using SQL Server</u> <u>Tools with Failover Clustering</u>.
- Replication. For more information, see <u>Creating a Failover Cluster</u>.
- SQL Profiler. For more information, see <u>Using SQL Server Tools with</u> <u>Failover Clustering</u>.

- SQL Query Analyzer. For more information, see <u>Using SQL Server</u> <u>Tools with Failover Clustering</u>.
- SQL Mail. For more information, see <u>Using SQL Server Tools with</u> <u>Failover Clustering</u>.

The following component is not supported for failover clustering:

• SQL Server 2000 Analysis Services

**Note** Microsoft Data Access Components (MDAC) 2.6 is not supported for SQL Server version 6.5 or SQL Server 7.0, when either version is in a failover cluster configuration.

Before using failover clustering, consider the following:

- Failover clustering resources, including the IP addresses and network name, must be used only when you are running an instance of SQL Server 2000. They should not be used for other purposes, such as file sharing.
- In a failover cluster configuration, SQL Server 2000 supports Windows NT 4.0, Enterprise Edition but requires that the service accounts for SQL Server services (SQL Server and SQL Server Agent) be local administrators of all nodes in the cluster.

**IMPORTANT** SQL Server 2000 supports both Named Pipes and TCP/IP Sockets over TCP/IP within a failover cluster. However, it is strongly recommended that you use TCP/IP Sockets in a clustered configuration.

# **Creating a Failover Cluster**

To create a Microsoft® SQL Server<sup>™</sup> 2000 failover cluster, you must create and configure the virtual servers on which the failover cluster runs. You create virtual servers during SQL Server Setup. Virtual servers are not provided by Microsoft Windows NT® 4.0 or Microsoft Windows® 2000.

To create a failover cluster, you must be a local administrator with rights to log on as a service and to act as part of the operating system on all computers in the failover cluster.

### **Elements of a Virtual Server**

A virtual server contains:

• A combination of one or more disks in a Microsoft Cluster Service (MSCS) cluster group.

Each MSCS cluster group can contain at most one virtual SQL Server.

- A network name for each virtual server. This network name is the virtual server name.
- One or more IP addresses that are used to connect to each virtual server.
- One instance of SQL Server 2000, including a SQL Server resource, a SQL Server Agent resource, and a full-text resource.

If an administrator uninstalls the instance of SQL Server 2000 within a virtual server, the virtual server, including all IP addresses and the network name, is also removed from the MSCS cluster group.

A failover cluster can run across one or more actual Windows 2000 Advanced Server or Windows 2000 Datacenter Server servers or Windows NT 4.0, Enterprise Edition servers that are participating nodes of the cluster. However, a SQL Server virtual server always appears on the network as a single Windows 2000 Advanced Server, Windows 2000 Datacenter Server, or Microsoft Windows NT 4.0, Enterprise Edition server.

### Naming a Virtual Server

SQL Server 2000 depends on distinct registry keys and service names within the failover cluster so that operations will continue correctly after a failover. Therefore, the name you provide for the instance of SQL Server 2000, including the default instance, must be unique across all nodes in the failover cluster, as well as across all virtual servers within the failover cluster. For example, if all instances failed over to a single server, their service names and registry keys would conflict. If INST1 is a named instance on virtual server VIRTSRV1, there cannot be a named instance INST1 on any node in the failover cluster, either as part of a failover cluster configuration or as a stand-alone installation.

Additionally, you must use the VIRTUAL\_SERVER\Instance-name string to connect to a clustered instance of SQL Server 2000 running on a virtual server. You cannot access the instance of SQL Server 2000 by using the computer name that the clustered instance happens to reside on at any given time. SQL Server 2000 does not listen on the IP address of the local servers. It listens only on the clustered IP addresses created during the setup of a virtual server for SQL Server 2000.

### **Usage Considerations**

Before you create a failover cluster, consider the following:

- If you are using the Windows 2000 Address Windowing Extensions (AWE) API to take advantage of memory greater than 3 gigabytes (GB), make certain that the maximum available memory you configure on one instance of SQL Server will still be available after you fail over to another node. If the failover node has less physical memory than the original node, instances of SQL Server may fail to start or may start with less memory than they had on the original node. You must:
  - Give each server in the cluster the same amount of physical RAM.
  - Ensure that the summed value of the **max server memory**

settings for all instances is less than the lowest amount of physical RAM available on any of the virtual servers in the failover cluster.

For more information about AWE, see <u>Using AWE Memory on</u> <u>Windows 2000</u>.

• If you need high-availability servers in replication, it is recommended that you use an MSCS cluster file share as your snapshot folder when configuring a Distributor on a failover cluster. In the case of server failure, the distribution database will be available and replication will continue to be configured at the Distributor.

Also, when creating publications, specify the MSCS cluster file share for the additional storage of snapshot files or as the location from which Subscribers apply the snapshot. This way, the snapshot files are available to all nodes of the cluster and to all Subscribers that must access it. For more information, see <u>Publishers, Distributors, and</u> <u>Subscribers</u> and <u>Alternate Snapshot Locations</u>.

- If you want to use encryption with a failover cluster, you must install the server certificate with the fully qualified DNS name of the virtual server on all nodes in the failover cluster. For example, if you have a two-node cluster, with nodes named test1.redmond.corp.microsoft.com and test2.redmond.corp.microsoft.com and a virtual SQL Server "Virtsql", you need to get a certificate for "virtsql.redmond.corp.microsoft.com" and install the certificate on both nodes. You can then check the Force protocol encryption check box on the Server Network Utility to configure your failover cluster for encryption.
- You should not remove the BUILTIN/Administrators account from SQL Server. The IsAlive thread runs under the context of the cluster service account, and not the SQL Server service account. The cluster service must be part of the administrator group on each node of the cluster. If you remove the BUILTIN/Administrators account, the IsAlive thread will no longer be able to create a trusted connection, and you will lose access to the virtual server.

### **Creating a Failover Cluster**

Here are the basic steps for creating a failover cluster using the Setup program:

1. Identify the information you need to create your virtual server (for example, cluster disk resource, IP addresses, and network name) and the nodes available for failover.

The cluster disks to use for failover clustering should all be in a single cluster group and owned by the node from which the Setup program is run. This configuration must take place before you run the Setup program. You configure this through Cluster Administrator in Windows NT 4.0 or Windows 2000. You need one MSCS group for each virtual server you want to set up.

2. Start the Setup program to begin your installation. After all necessary information has been entered, the Setup program installs a new instance of SQL Server binaries on the local disk of each computer in the cluster and installs the system databases on the specified cluster disk. The binaries are installed in exactly the same path on each cluster node, so you must ensure that each node has a local drive letter in common with all the other nodes in the cluster.

In SQL Server 2000, during a failover only the databases fail over. In SQL Server version 6.5 and SQL Server version 7.0, both the SQL Server databases and binaries fail over during a failover.

If any resource (including SQL Server) fails for any reason, the services (SQL Server, SQL Server Agent, Full-Text Search, and all services in the failover cluster group) fail over to any available nodes defined in the virtual server.

3. You install one instance of SQL Server 2000, creating a new virtual server and all resources.

#### How to create a new failover cluster

# **Failover Clustering Example**

The following example illustrates how you configure Microsoft® SQL Server<sup>™</sup> 2000 failover clustering.

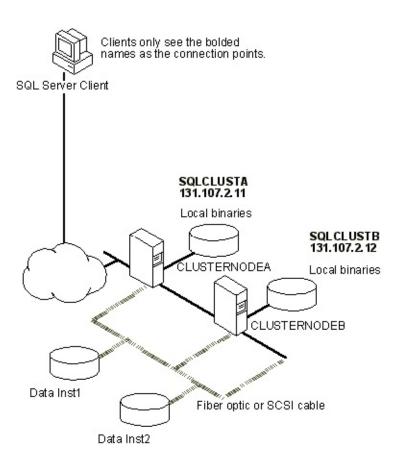
CLUSTERNODEA and CLUSTERNODEB are two computers in a failover cluster. Run SQL Server Setup on CLUSTERNODEA and create a virtual server named "SQLCLUSTA." Then install a default instance of SQL Server 2000, which can run on both CLUSTERNODEA and CLUSTERNODEB. From this point forward, connect to the server by specifying "SQLCLUSTA" as the server name in the connection string.

Run the Setup program again on CLUSTERNODEB. Create a new virtual server named "SQLCLUSTB" (in a different Microsoft Cluster Service (MSCS) cluster group) and install an instance named "Inst1" that can run on both CLUSTERNODEA and CLUSTERNODEB. From this point forward, connect to the server by specifying "SQLCLUSTB\Inst1" as the connection string.

The two virtual servers are running in the MSCS cluster consisting of CLUSTERNODEA and CLUSTERNODEB. Other than that, they are completely separate from each other. Each virtual server resides in a different MSCS cluster group, and each has a different set of IP addresses, a distinct network name, and data files that reside on a separate set of shared cluster disks.

When a failover occurs for any resource in an MSCS cluster group, all resources that are members of that group also fail over. For SQLCLUSTA, any failure (from the disk resources, IP address, the network name, or the installations of SQL Server 2000 within the virtual server) causes all members of the cluster group to fail over when the failover threshold is reached.

The following illustration is a two-node cluster with binaries and data. Each virtual server in this illustration must have exclusive ownership of the disk on which the data and log files are located.



## See Also

Failover Clustering Architecture

# **Upgrading to a SQL Server 2000 Failover Cluster**

When you are upgrading to a Microsoft® SQL Server<sup>™</sup> 2000 failover cluster, only one default instance is allowed. Use the Cluster Wizard in SQL Server version 6.5 or SQL Server 7.0 to uncluster any existing SQL Server 6.5 or SQL Server 7.0 clustered instances before upgrading to SQL Server 2000. Then run SQL Server Setup on SQL Server 2000.

SQL Server 6.5 or SQL Server 7.0 failover clusters cannot exist on the same computer as a SQL Server 2000 failover cluster. In SQL Server 6.5 or SQL Server 7.0, in an active/active configuration or in an active/passive configuration where one server contains an unclustered SQL Server, there is a name conflict. Both servers are default instances.

**IMPORTANT** You cannot run the Cluster Wizard in SQL Server 6.5 or SQL Server 7.0 after SQL Server 2000 has been installed.

For SQL Server 2000, you must use a domain account for the services (SQL Server, SQL Server Agent, and all services in the clustered group). That account must be an administrator on all computers in the cluster, if those computers are running on Microsoft Windows NT® Server 4.0, Enterprise Edition.

**Note** If you are using replication on a SQL Server 6.5 or 7.0 failover cluster and upgrading to a SQL Server 2000 failover cluster, you must uncluster the previous installation. Delete all publications, remove replication, and then reconfigure replication after upgrading. This will not be a requirement when upgrading from SQL Server 2000 in future releases.

#### To upgrade from a SQL Server 6.5 active/passive failover cluster

## Handling a Failover Cluster Installation

When you install a Microsoft<sup>®</sup> SQL Server<sup>™</sup> 2000 failover cluster, you must:

- Ensure that the operating system is installed properly and designed to support failover clustering. For more information about what to do before installing a failover cluster, see <u>Before Installing Failover</u> <u>Clustering</u>. For more information about the order of installation, see <u>Installing Failover Clustering</u>.
- Consider whether the SQL Server tools, features, and components you want to use are supported with failover clustering. For more information, see <u>Failover Clustering Support</u>.
- Consider whether failover clustering is dependent on the products you want to use. For more information, see <u>Failover Clustering</u> <u>Dependencies</u>.
- Consider how to create a new failover cluster. For more information about creating a new failover cluster configuration, see <u>Creating a Failover Cluster</u>.
- Review the instructions for upgrading from a SQL Server version 6.5 or SQL Server version 7.0 cluster to a SQL Server 2000 failover cluster. For more information, see <u>Upgrading to a SQL Server 2000 Failover</u> <u>Cluster</u>.

# **Before Installing Failover Clustering**

Before you install a Microsoft® SQL Server<sup>™</sup> 2000 failover cluster, you must select the operating system on which your computer will run. You can use Microsoft Windows NT® 4.0, Enterprise Edition, Microsoft Windows® 2000 Advanced Server, or Microsoft Windows 2000 Datacenter Server. You also must install Microsoft Cluster Service (MSCS).

#### **Preinstallation Checklist**

Before you begin the installation process, verify that:

- There is no IRQ sharing between network interface cards (NICs) and drive/array (SCSI) controllers. Although some hardware may support this sharing, it is not recommended.
- Your hardware is listed on the Windows NT Hardware Compatibility List.

For a complete list of supported hardware, see the Hardware Compatibility List at the <u>Microsoft Web site</u>.

The hardware system must appear under the category of cluster. Individual cluster components added together do not constitute an approved system. Only systems purchased as a cluster solution and listed in the cluster group are approved. When checking the list, specify cluster as the category. All other categories are for OEM use.

• MSCS has been installed completely on at least one node before you run Windows NT 4.0, Enterprise Edition or Windows 2000 Advanced Server or Windows 2000 Datacenter Server simultaneously on all nodes.

When using MSCS, you must make certain that one node is in control of the shared SCSI bus prior to the other node(s) coming online. Failure to do this can cause application failover to go into an online pending state. As a result, the cluster either fails on the other node or fails totally. However, if your hardware manufacturer has a proprietary installation process, follow the hardware manufacturer instructions.

• WINS is installed according to the following article in the Product Support Services <u>Microsoft Web site</u>:

Q258750 Recommended Private "Heartbeat" Configuration on Cluster Server

- The disk drive letters for the cluster-capable disks are the same on both servers.
- You have disabled NetBIOS for all private network cards before beginning SQL Server Setup.
- You have cleared the system logs in all nodes and viewed the system logs again. Ensure that the logs are free of any error messages before continuing.

# **Installing Failover Clustering**

If you are installing Microsoft® SQL Server<sup>™</sup> 2000 failover clustering on Microsoft Windows NT® 4.0, Enterprise Edition, you need to install programs in the order specified below. However, this is not necessary if you are installing failover clustering on Microsoft Windows® 2000 Advanced Server or Windows 2000 Datacenter Server.

**CAUTION** If you do not install the programs in the following order, the software products can fail on installation and require that you completely reinitialize the disk and restart installation.

Before installing SQL Server 2000 in a failover cluster configuration, you must upgrade any pre-release versions of SQL Server 2000.

#### To install failover clustering on Windows NT 4.0

1. Install Windows NT 4.0, Enterprise Edition.

Windows NT 4.0, Enterprise Edition includes Windows NT 4.0 Service Pack 3. Service Pack 3 is required to install Microsoft Cluster Service (MSCS).

- Do not go directly to Service Pack 4 or later if you intend to install the Windows NT Option Pack.
- Do not install Microsoft Internet Information Server (IIS).

**IMPORTANT** IIS is installed by default. It is recommended that you clear this option during the Windows NT 4.0 installation.

- 2. Install MSCS.
- 3. Install Microsoft Internet Explorer version 5.0 or later.
- 4. Manually create a Microsoft Distributed Transaction Coordinator (MS DTC) compatible resource group where MS DTC setup can create its

resources. This should contain an IP address, network name, and cluster disk resource. Any group with these three things is compatible with MS DTC.

SQL Server Setup will install MS DTC in a later step. Install Windows NT 4.0 Option Pack only if you require components of the Windows NT 4.0 Option pack besides MS DTC.

5. Install the latest Windows NT 4.0 Service Pack, Service Pack 5 at the latest. Click **Create an uninstall directory**, click **Year 2000 Setup**, and then select the **Service Pack install for Intel based systems** check box.

Do not select Microsoft Message Queue Server (MSMQ 1.0) or IIS. MSMQ 1.0 is not supported on SQL Server 2000. It is recommended that IIS functionality be used with Windows NT Load Balancing Service (WLBS). For more information about WLBS, search on "WLBS Features Overview" on the NT Server <u>Microsoft Web site</u>.

Prior to Step 5, it is recommended that you rename the hidden directory \$NTServicePackUninstall\$ to \$NTServicePackUninstall\$.*service packnumber*. After installing the service pack, add a new directory. This way you have uninstall directories available, which prevents the directories from being accidentally overwritten.

6. Install SQL Server 2000.

**Note** Install any additional server products before installing any other applications.

#### To install failover clustering on Windows 2000

- 1. Install Windows 2000 and accept the default application choices.
- 2. After installing Windows 2000 on the first node and prior to installing MSCS, click **Start\Programs\Administrative Tools\Configure Your Server**.

- 3. Click **Advanced**\**Cluster Service**, and then in the right pane, click **Learn More**.
- 4. From **Help**, review Item 2 under Windows Clustering.

Windows Clustering is used during the installation of Windows 2000 and with SQL Server 2000 failover clustering. Follow these instructions to install MSCS.

**IMPORTANT** It is necessary to read the section on Planning for Windows Clustering\Requirements for server clusters and to follow the Checklist for server clusters called Checklist: Creating a server cluster. This is found under the Server Clusters section\Checklist for server clusters.

5. After you have successfully installed MSCS, you need to configure MS DTC to run on a cluster.

For more information about MS DTC, see <u>Failover Clustering</u> <u>Dependencies</u>.

- 6. On the **Start** menu, point to **Programs\Administrative Tools\Cluster Administrator**, and click **View Groups\Cluster Group**. If the group contains an MS DTC resource, proceed to Step 9. If not, complete the following two steps.
- 7. On the **Start** menu, point to **Command Prompt**. Enter comclust.exe from the command prompt.
- 8. Repeat Step 7 on the remaining nodes of the cluster, one node at a time.
- 9. Install SQL Server 2000.

**Note** Install any additional server products before installing any user applications.

SQL Server Setup Help

### **Failover Clustering Dependencies**

There are several products that interact with Microsoft® SQL Server<sup>™</sup> 2000 failover clustering. To ensure that your failover cluster functions properly, you need to understand the underlying dependencies that failover clustering has on other products.

#### **Microsoft Distributed Transaction Coordinator (MS DTC)**

SQL Server 2000 requires Microsoft Distributed Transaction Coordinator (MS DTC) in the cluster for distributed queries and two-phase commit transactions, as well as for some replication functionality. After you install Microsoft Windows® 2000 and configure your cluster, you must run the Cluster Wizard (the comclust.exe program) on all nodes to configure MS DTC to run in clustered mode.

The Cluster Wizard makes the following changes to the MS DTC configuration:

- It creates an MS DTC resource in a resource group containing a shared cluster disk resource and a network name resource.
- It creates an MS DTC log file on the shared cluster disk contained in the MS DTC resource group. Placing the MS DTC log file on the shared cluster disk makes it possible for the MS DTC transaction manager to access the MS DTC log file from any system in the cluster.
- It copies critical MS DTC registry entries to the shared cluster registry.

#### **Running MS DTC in Clustered Mode**

When MS DTC is running in clustered mode, only one node in the cluster runs the MS DTC transaction manager at a time.

Any process running on any node in the cluster can use MS DTC. These processes simply call the MS DTC Proxy and the MS DTC Proxy automatically forwards MS DTC calls to the MS DTC transaction manager that is controlling

the entire cluster.

If the node running the MS DTC transaction manager fails, the MS DTC transaction manager is automatically restarted on another node in the cluster. The newly restarted MS DTC transaction manager reads the MS DTC log file on the shared cluster disk to determine the outcome of pending and recently completed transactions. Resource managers reconnect to the MS DTC transaction manager and perform recovery to determine the outcome of in-doubt transactions. Applications reconnect to MS DTC so they can initiate new transactions.

For example, suppose the MS DTC transaction manager is active on system B. The application program and resource manager on system A call the MS DTC proxy. The MS DTC proxy on system A forwards all MS DTC calls to the MS DTC transaction manager on system B.

If system B fails, the MS DTC transaction manager on system A will take over. It will read the entire MS DTC log file on the shared cluster disk, perform recovery, and then serve as the transaction manager for the entire cluster.

**Note** The MS DTC transaction manager, MS DTC Proxy, and Component Services administrative tools are installed on each node of a Windows 2000 cluster using MSCS as part of Windows 2000 Setup.

#### To manually install MS DTC on a Windows 2000 system running MSCS

- 1. Install Windows 2000 on each node in the cluster.
- 2. Use the Windows 2000 Configure Your Server facility to configure your cluster.
- 3. From a command prompt, run comclust.exe on each node in the cluster. Comclust.exe can be found in the system32 directory.

#### To automatically install MS DTC on a Windows 2000 cluster system

- 1. Install Windows 2000 on each node in the cluster and configure your cluster using automatic installation scripts.
- 2. From a command prompt, run comclust.exe on each node in the

cluster. Comclust.exe can be found in the system32 directory.

## To upgrade a non-clustered Windows NT 4.0 SP4 system to a Windows 2000 cluster

- 1. Upgrade each system that will be part of the cluster to Windows 2000.
- 2. Use the Windows 2000 Configure Your Server facility to configure your server.
- 3. From a command prompt, run comclust.exe on each node in the cluster. Comclust.exe can be found in the system32 directory.

## To upgrade a clustered Windows NT 4.0 SP4 system to a Windows 2000 cluster

1. Install Windows 2000 on each node in the cluster.

MS DTC requires that all nodes in the cluster be upgraded to Windows 2000 at the same time.

2. From a command prompt, run comclust.exe on each node in the cluster. Comclust.exe can be found in the system32 directory.

**IMPORTANT** Microsoft System Management Server 1.2 is not supported with SQL Server or Microsoft Cluster Service (MSCS).

## To recover from a cluster failure and rebuild MS DTC on a Windows 2000 cluster

- 1. When a node is lost, MS DTC will continue to work on the remaining nodes in the cluster. It does not matter whether the node that is lost is the primary or secondary node.
- 2. When you are ready to restore the lost node, join the lost node back to the cluster. After the node has joined the cluster, run Comclust.exe, which can be found in the system32 directory. This will reconfigure MS DTC on the node.

SQL Server Setup Help

### **Maintaining a Failover Cluster**

After you have installed a Microsoft® SQL Server<sup>™</sup> 2000 failover cluster, you can change or repair your existing setup. For example, you can add additional nodes to a virtual server in a failover cluster, run a clustered instance as a standalone instance, remove a node from a clustered instance, or recover from failover cluster failure.

#### Adding a Node to an Existing Virtual Server

During SQL Server Setup, you are given the option of maintaining an existing virtual server. If you choose this option, you can add other nodes to your failover cluster configuration at a later time. You can add up to three additional nodes to an existing virtual server configured to run on one node.

#### To add a node to an existing virtual server

SQL Server Setup Help

### **Using SQL Server Tools with Failover Clustering**

You can use Microsoft® SQL Server<sup>™</sup> 2000 failover clustering with a variety of SQL Server tools and features. However, review the following usage considerations.

#### **Full-Text Queries**

To use the Microsoft Search service to perform full-text queries with failover clustering, consider the following:

- An instance of SQL Server 2000 must run on the same system account on all failover cluster nodes in order for full-text queries to work on failover clusters.
- You must change the start-up account for SQL Server 2000 in the failover cluster using SQL Server Enterprise Manager. If you use Control Panel or the Services Application in Microsoft Windows® 2000, you will break the full-text configuration for SQL Server.

#### **SQL Server Enterprise Manager**

To use SQL Server Enterprise Manager with failover clustering, consider the following:

- You must change the start-up account for SQL Server 2000 in the failover cluster by using SQL Server Enterprise Manager. If you use Control Panel or the Services Application in Microsoft Windows 2000, you could break your server configuration.
- When creating or altering databases, you will only be able to view the cluster disks for the local virtual server.
- If you are browsing a table through SQL Server Enterprise Manager and lose the connection to SQL Server during a failover, you will see the

error message, "Communication Link Failure". You must press ESC and undo the changes to exit out of the SQL Server Enterprise Manager window. You cannot click **Run Query**, save any changes, or edit the grid.

• If you use Enterprise Manager to reset the properties of the SQL Server service account, you will be prompted to restart SQL Server. When SQL Server is running in a failover cluster configuration, this will bring the full text and SQL Agent resources offline, as well as SQL Server. However, when SQL Server is restarted, it will not bring the full text or SQL Agent resources back online. You must start those resources manually using the Windows Cluster Administrator utility.

#### **Service Control Manager**

Use the Service Control Manager to start or stop a clustered instance of SQL Server. You cannot pause a clustered instance of SQL Server.

To start a clustered instance of SQL Server using Service Control Manager

SQL Server Setup Help

### **Failover Cluster Troubleshooting**

This topic provides information about:

- Resolving the most common Microsoft® SQL Server<sup>™</sup> 2000 failover clustering usage issues.
- Optimizing failover cluster performance.
- Using failover clustering with extended stored procedures that use COM objects.

#### **Resolving Common Usage Issues**

The following list describes common usage issues and explains how to resolve them:

• SQL Server 2000 cannot log on to the network after it migrates to another node.

SQL Server service account passwords must be identical on all nodes or else the node cannot restart a SQL Server service that has migrated from a failed node.

If you change the SQL Server service account passwords on one node, you must change the passwords on all other nodes. However, if you change the account using SQL Server Enterprise Manager, this task will be done automatically.

• SQL Server cannot access the cluster disks.

A node cannot recover cluster disks that have migrated from a failed node if the shared cluster disks use a different letter drive. The disk drive letters for the cluster disks must be the same on both servers. If they are not, review your original installation of the operating system and Microsoft Cluster Service (MSCS). For more information, see the Microsoft Windows NT® 4.0, Enterprise Edition, Windows® 2000 Advanced Server, or Windows 2000 Datacenter Server documentation.

• You do not want a failure of a service, such as full-text search or SQL Server Agent, to cause a failover.

To prevent the failure of specific services from causing the SQL Server group to fail over, configure those services using Cluster Administrator in Windows NT 4.0 or Windows 2000. For example, to prevent the failure of the Full-Text Search service from causing a failover of SQL Server, clear the **Affect the Group** check box on the **Advanced** tab of the **Full Text Properties** dialog box. However, if SQL Server causes a failover, the full-text search service will restart.

• SQL Server will not start automatically.

You cannot start a failover cluster automatically using SQL Server. You must use Cluster Administrator in MSCS to automatically start a failover cluster.

• The error message "No compatible resource groups found" is displayed during SQL Server Setup.

This error is caused by the Microsoft Distributed Transaction Coordinator (MS DTC) setup on Windows NT 4.0, Enterprise Edition. MS DTC requires a group containing a network name, IP address, and shared cluster disk to be owned by the local node when the Setup program is run. If this error is displayed, open Cluster Administrator and make certain there is a group that meets these requirements owned by the local node. The easiest way to do this is to move a disk into the cluster group that already contains a network name and IP address. After you have this group on the local node, click **Retry**.

• The error message "All cluster disks available to this virtual server are owned by other node(s)" is displayed during Setup.

This message is displayed when you select the drive and path for installing data files, and the drive you selected is not owned by the local node. Move the disk to the local node using Cluster Administrator.

• The error message "Unable to delete SQL Server resources. They must be manually removed. Uninstallation will continue." is displayed during

SQL Server Setup.

This message is displayed if SQL Server Setup cannot delete all of the SQL Server resources. You must go into Control Panel and uninstall the instance you were trying to remove on every node.

• You cannot enable the clustering operating system error log.

The operating system cluster error log is used by MSCS to record information about the cluster. Use this error log to debug cluster configuration issues. To enable the cluster error log, set the system environment variable CLUSTERLOG=cpath to file> (for example, CLUSTERLOG=c:\winnt\cluster\cluster.log). This error log is on by default in Windows 2000.

• If the Network Name is offline and you cannot connect using TCP/IP, you must use Named Pipes.

To connect using Named Pipes, create an alias using the Client Network Utility to connect to the appropriate computer. For example, if you have a cluster with two nodes (Node A and Node B), and a virtual server (Virtsql) with a default instance, you can connect to the server that has the Network Name resource offline by doing the following:

- 1. Determine on which node the group containing the instance of SQL Server is running by using the Cluster Administrator. For this example, it will be Node A.
- 2. Start the SQL Server service on that computer using **net start**. For more information about using **net start**, see <u>Starting SQL</u> <u>Server Manually</u>.
- 3. Start the SQL Server Network Utility on Node A. View the pipe name on which the server is listening. It should be similar to \\.\\$\$\VIRTSQL\pipe\sql\query.
- 4. On the client computer, start the Client Network Utility.

5. Create an alias SQLTEST1 to connect via Named Pipes to this pipe name. To do this, put Node A as the server name and edit the pipe to be \\.\pipe\\$\$\VIRTSQL\sql\query. Connect to this instance using the alias SQLTEST1 as the server name.

For more information, see <u>Client Net-Libraries and Network Protocols</u>.

#### **Optimizing Failover Clustering Performance**

To optimize performance when using failover clustering, consider the following:

- If your disk controller is not external to your clustered computer, you must turn off write-caching within the controller to prevent data loss during a failover.
- Write-back caching cannot be used on host controllers in a cluster without hindering performance. However, if you use external controllers, you continue to provide performance benefits. External disk arrays are not affected by failover clustering and can sync the cache correctly, even across a SCSI bus.
- It is recommended that you do not use the cluster drive for file shares. Using these drives impacts recovery times and can cause a failover of the cluster group due to resource failures.

#### **Using Extended Stored Procedures and COM Objects**

When you use extended stored procedures with a failover clustering configuration, all extended stored procedures need to be installed on the shared cluster disk. This is to ensure that when a node fails over, the extended stored procedures can still be used.

If the extended stored procedures use COM components, the administrator needs to register the COM components on each node of the cluster. The information for loading and executing COM components must be in the registry of the active node in order for the components to be created. Otherwise, the information will remain in the registry of the computer on which the COM components were first registered. For more information, see <u>Extended Stored Procedure Architecture</u>.

SQL Server Setup Help

### How to create a new failover cluster (Setup)

**IMPORTANT** Before you create a Microsoft® SQL Server<sup>TM</sup> 2000 failover cluster, you must configure Microsoft Cluster Service (MSCS) and use Cluster Administrator in Microsoft Windows NT® 4.0 or Windows® 2000 to create at least one cluster disk resource. Note the location of the cluster drive in the Cluster Administrator before you run SQL Server Setup because you need this information to create a new failover cluster.

#### To create a new failover cluster

- 1. On the **Welcome** screen of the **Microsoft SQL Server Installation Wizard**, click **Next**.
- 2. On the **Computer Name** screen, click **Virtual Server** and enter a virtual server name. If Setup detects that you are running MSCS, it will default to **Virtual Server**. Click **Next**.
- 3. On the **User Information** screen, enter the user name and company. Click **Next**.
- 4. On the **Software License Agreement** screen, click **Yes**.
- 5. On the **Failover Clustering** screen, enter one IP address for each network configured for client access. That is, enter one IP address for each network on which the virtual server will be available to clients on a public (or mixed) network. Select the network for which you want to enter an IP address, and then enter the IP address. Click **Add**.

The IP address and the subnet are displayed. The subnet is supplied by MSCS. Continue to enter IP addresses for each installed network until you have populated all desired networks with an IP address. Click **Next**.

- 6. On the **Cluster Disk Selection** screen, select the cluster disk group where the data files will be placed by default. Click **Next**.
- 7. On the **Cluster Management** screen, review the cluster definition provided by SQL Server 2000. By default, all available nodes are selected. Remove any nodes that will not be part of the cluster definition for the virtual server you are creating. Click **Next**.
- 8. On the **Remote Information** screen, enter login credentials for the remote cluster node. The login credentials must have administrator privileges on the remote node(s) of the cluster. Click **Next**.
- 9. On the **Instance Name** screen, choose a default instance or specify a named instance. To specify a named instance, clear the **Default** check box, and then enter the name for the named instance. Click **Next**.

**IMPORTANT** You cannot name an instance DEFAULT or MSSQLSERVER. For more information about naming instances of SQL Server 2000, see <u>Working with Named and Multiple Instances of</u> <u>SQL Server 2000</u>. Names must follow rules for SQL Server identifiers. For more information about naming conventions for identifiers, see <u>Using Identifiers</u>.

10. On the **Setup Type** screen, select the type of installation to install. The Setup program automatically defaults to the first available cluster disk resource from the group you previously selected.

However, if you need to specify a different clustered drive resource, under **Data Files**, click **Browse** and then specify a path on a clustered drive resource. You will be required to select a clustered drive resource that is owned by the node on which you are running the Setup program. The drive also must be a member of the cluster group you previously selected. Click **Next**.

11. On the **Services Accounts** screen, select the service account(s) that you want to run in the failover cluster. Click **Next**.

- 12. In the Authentication Mode dialog box, choose the authentication mode to use. If you change the selection from Windows Authentication Mode to Mixed Mode (Windows Authentication and SQL Server Authentication), you need to enter and confirm a password for the sa login.
- 13. On the **Start Copying Files** screen, click **Next**.
- 14. On the **Setup Complete** screen, click **Finish**.

If you are instructed to restart the computer, do so now. It is important to read the message from the Setup program when you are done with installation. Failure to restart any of the specified nodes may cause failures when you run the Setup program in the future on any node in the failover cluster.

### How to install a one-node failover cluster (Setup)

- 1. On the **Welcome** screen of the **Microsoft SQL Server Installation Wizard**, click **Next**.
- 2. On the **Computer Name** screen, click **Virtual Server** and enter a virtual server name. If SQL Server Setup detects that you are running Microsoft® Cluster Service (MSCS), it will default to **Virtual Server**. Click **Next**.
- 3. On the **User Information** screen, enter the user name and company. Click **Next**.
- 4. On the **Software License Agreement** screen, click **Yes**.
- 5. On the **Failover Clustering** screen, enter one IP address per installed network for the virtual server. Select the network for which you wish to enter an IP address, and then enter the IP address. Click **Add**.

The IP address and the subnet are displayed. The subnet is supplied by MSCS. Continue to enter IP addresses for each installed network until you have populated all desired networks with an IP address. Click **Next**.

- 6. On the **Cluster Disk Selection** screen, select the cluster disk group where the data files will be placed by default. Click **Next**.
- 7. On the **Cluster Management** screen, review the failover cluster definition provided by Microsoft SQL Server<sup>™</sup> 2000. By default, all available nodes are selected. Remove any nodes that will not be part of the cluster definition for the virtual server you are creating. Click **Next**.

- 8. On the **Remote Information** screen, enter login credentials that have administrator privileges on the remote node of the cluster. Click **Next**.
- 9. On the **Instance Name** screen, choose a default instance or specify a named instance. To specify a named instance, clear the **Default** check box, and then enter the name. Click **Next**.

**IMPORTANT** You cannot name an instance DEFAULT or MSSQLSERVER. The name must follow the rules for SQL Server identifiers. For more information about naming conventions for identifiers, see <u>Using Identifiers</u>.

- 10. On the **Setup Type** screen, select the type of installation to install. Setup will automatically default to the first available clustered disk resource from the group you previously selected. However, if you need to specify a different clustered drive resource, under **Data Files**, click the **Browse** button and then specify a path on a clustered drive resource. You will be required to select a clustered drive resource that is owned by the node on which you are running Setup. The drive must also be a member of the cluster group you previously selected. Click **Next**.
- 11. On the **Services Accounts** screen, select the service account(s) that you want to run in the failover cluster. Click **Next**.
- 12. In the Authentication Mode dialog box, choose the authentication mode to use. If you change the selection from Windows Authentication Mode to Mixed Mode (Windows Authentication and SQL Server Authentication), you must enter and confirm a password for the sa login.
- 13. On the **Start Copying Files** screen, click **Next**.
- 14. On the **Setup Complete** screen, click **Finish**. If you are instructed to restart the computer, do so now. It is important to read the message

from the Setup program when you are done with installation. Failure to restart any of the specified nodes may cause failures when running the Setup program in the future on any node in the cluster.

### How to add nodes to an existing virtual server (Setup)

- 1. On the **Welcome** screen of the **Microsoft SQL Server Installation Wizard**, click **Next**.
- 2. On the **Computer Name** screen, click **Virtual Server** and specify the virtual server to which you want to add a node. Click **Next**.
- 3. On the **Installation Selection** screen, click **Advanced options**. Click **Next**.
- 4. On the **Advanced Options** screen, click **Maintain a virtual server for failover clustering**. Click **Next**.
- 5. On the **Failover Clustering** screen, click **Next**.

You do not need to enter an IP address.

6. On the **Cluster Management** screen, select the node and click **Add**.

If the node is listed as unavailable, you must modify the disk resources in the cluster group of the virtual server so the disk is available for the node you want to add to the Microsoft® SQL Server<sup>™</sup> configuration. Click **Next**.

- 7. On the **Remote Information** screen, enter login credentials for the remote cluster node that has administrator privileges on the remote node of the cluster. Click **Next**.
- 8. On the **Setup Complete** screen, click **Finish**.

# How to remove a node from an existing failover cluster (Setup)

- 1. On the **Welcome** screen of the **Microsoft SQL Server Installation Wizard**, click **Next**.
- 2. On the **Computer Name** screen, click **Virtual Server** and specify the name of the server from which to remove the node. Click **Next**.
- 3. You may see an error message saying that one (or more) of the nodes of the Microsoft® Windows NT® 4.0 or Microsoft Windows® 2000 cluster are unavailable. This may be because the node(s) you are attempting to remove is damaged. The node(s) still can be removed. Click **OK**.
- 4. On the **Installation Selection** screen, click **Advanced Options**. Click **Next**.
- 5. On the **Advanced Options** screen, click **Maintain a virtual server for failover clustering**. Click **Next**.
- 6. On the **Failover Clustering** screen, click **Next**.

You do not need to modify any IP address(es).

- 7. On the **Cluster Management** screen, select the node and click **Remove**. Click **Next**.
- 8. On the **Remote Information** screen, enter login credentials for the remote cluster node that has administrator privileges on the remote node(s) of the cluster. Click **Next**.

9. On the **Setup Complete** screen, click **Finish**.

If you are instructed to restart the computer, do so now. It is important to read the message from SQL Server Setup when you are done with installation. Failure to restart any of the specified nodes may cause failures when you run the Setup program in the future on any node in the failover cluster.

### How to remove a failover clustered instance (Setup)

- 1. On the **Welcome** screen of the **Microsoft SQL Server Installation Wizard**, click **Next**.
- 2. On the **Computer Name** screen, click **Virtual Server** and specify the name of the server from which to remove a clustered instance. Click **Next**.
- 3. On the **Installation Selection** screen, click **Upgrade, remove, or add components to an existing instance of SQL Server**.
- 4. On the **Instance Name** screen, for a default instance, click **Default**. For a named instance, specify the name of the instance to remove. Click **Next**.
- 5. On the **Existing Installation** screen, click **Uninstall your existing installation**. Click **Next**.
- 6. On **the Remote Information** screen, specify the password that is a valid administrator password on all nodes in the cluster. Click **Next**.
- 7. In the **Setup** message "Successfully uninstalled the instance . . . ", click **OK**.
- 8. On the **Setup Complete** screen, click **Finish**.

If you are instructed to restart the computer, do so now. It is important to read the message from SQL Server Setup when you are done with installation. Failure to restart any of the specified nodes may cause failures when you run the Setup program in the future on any node in the failover cluster.

SQL Server Setup Help

# How to recover from failover cluster failure in Scenario 1

In this scenario, failure is caused by hardware failure in Node 1 of a two-node cluster. This hardware failure could be caused, for example, by the failure of a small computer system interface (SCSI) card or the operating system.

- 1. After Node 1 fails, the Microsoft® SQL Server<sup>™</sup> 2000 failover cluster fails over to Node 2.
- 2. Run SQL Server Setup and remove Node 1. For more information, see <u>How to remove a failover clustered instance</u>.
- 3. Evict Node 1 from Microsoft Cluster Service (MSCS). To evict a node from MSCS, from Node 2, right-click on the node to remove, and then click **Evict Node**.
- 4. Install new hardware to replace the failed hardware in Node 1.
- 5. Install the operating system. For more information about which operating system to install and specific instructions on how to do this, see <u>Before Installing Failover Clustering</u>.
- 6. Install MSCS and join the existing cluster. For more information, see <u>Before Installing Failover Clustering</u>.
- 7. Run the Setup program on Node 2 and add Node 1 back to the failover cluster. For more information, see <u>How to add nodes to an existing</u> <u>virtual server (Setup)</u>.

SQL Server Setup Help

# How to recover from failover cluster failure in Scenario 2

In Scenario 2, failure is caused by Node 1 being down or offline but not irretrievably broken. This could be caused, for example, by an operating system failure.

- 1. After Node 1 fails, the Microsoft® SQL Server<sup>™</sup> 2000 failover cluster fails over to Node 2.
- 2. Run SQL Server Setup and remove Node 1. For more information, see <u>How to remove a failover clustered instance</u>.
- 3. Resolve the problem with Node 1.
- 4. Ensure that the Microsoft Cluster Service (MSCS) cluster is working and all nodes are online.
- 5. Run the Setup program on Node 2 and add Node 1 back to the failover cluster. For more information, see <u>How to add nodes to an existing</u> <u>virtual server (Setup)</u>.

# How to upgrade from a SQL Server 6.5 active/passive failover cluster (Setup)

#### To upgrade from a SQL Server 6.5 active/passive failover cluster

- 1. Uncluster Microsoft<sup>®</sup> SQL Server<sup>™</sup> version 6.5.
- 2. Install a default instance of SQL Server 2000.

You must install the binaries to a local drive and use a cluster disk for the data. This local drive is a path, which is a non-clustered disk valid on all nodes of the cluster. On all nodes of the cluster, this drive must have at least 300 megabytes (MB) of available space.

- 3. Run the SQL Server Upgrade Wizard to migrate your data into SQL Server 2000.
- 4. Uninstall SQL Server 6.5.
- 5. Run SQL Server Setup to upgrade your default instance of SQL Server 2000 to a SQL Server 2000 failover cluster.

For more information, see <u>How to upgrade from a default instance to a</u> <u>default clustered instance of SQL Server 2000 (SQL Server Setup)</u>.

#### See Also

<u>How to perform a SQL Server version 6.5 to SQL Server 2000 upgrade using a</u> <u>direct pipeline (SQL Server Upgrade Wizard)</u>

How to perform a SQL Server version 6.5 to SQL Server 2000 upgrade using a tape drive (SQL Server Upgrade Wizard)

# How to upgrade from a SQL Server 6.5 active/active failover cluster (Setup)

**Note** To upgrade from a Microsoft® SQL Server<sup>™</sup> 6.5 active/active failover cluster (or any configuration where SQL Server exists on the second node), you must first convert one side of the failover cluster to a named instance of SQL Server 2000.

#### To upgrade from a SQL Server 6.5 active/active failover cluster

- 1. On Node 1, uncluster SQL Server 6.5. On Node 2, uncluster SQL Server 6.5.
- 2. On Node 1, install a default (non-clustered) instance of SQL Server 2000.

You must install the binaries to a local drive and use a cluster disk for the data. This local drive is a path, which is a non-clustered disk valid on all nodes of the cluster. This drive on all nodes of the cluster must have at least 300 megabytes (MB) of available space.

- 3. On Node 1, run the SQL Server 2000 Upgrade Wizard to migrate your data into SQL Server 2000.
- 4. On Node 1, uninstall the instance of SQL Server 6.5.
- 5. On Node1, install a named, clustered instance of SQL Server 2000.
- 6. Run the Copy Database Wizard (CDW.exe) to migrate your SQL Server data (originally from SQL Server 6.5) to a named instance in a SQL Server 2000 failover cluster. For more information about the Copy Database Wizard, see <u>Using the Copy Database Wizard</u> or <u>How</u> to upgrade databases online using the Copy Database Wizard (Enterprise Manager).

- 7. On Node 1, uninstall the default instance of SQL Server 2000.
- 8. On Node 2, install a default instance of SQL Server 2000.
- 9. Run the SQL Server 2000 Upgrade Wizard to migrate your data into SQL Server 2000.

You must install the binaries to a local drive and use a cluster disk for the data. This local drive is a path, which is a non-clustered disk valid on all nodes of the cluster. On all nodes of the cluster, this drive must have at least 300 megabytes (MB) of available space.

- 10. On Node 2, uninstall the instance of SQL Server 6.5.
- 11. On Node 2, upgrade the default instance of SQL Server to a clustered instance.

For more information, see <u>How to upgrade from a default instance to a</u> <u>default clustered instance of SQL Server 2000 (SQL Server Setup)</u>.

# How to upgrade from a SQL Server 7.0 active/active failover cluster (Setup)

**Note** To upgrade from a Microsoft® SQL Server<sup>™</sup> version 7.0 active/active failover cluster (or any configuration where SQL Server exists on the second node), you must first convert one side of the failover cluster to a named instance of SQL Server 2000.

#### To upgrade from a SQL Server 7.0 active/active failover cluster

- 1. On Node 1, uncluster SQL Server version 7.0. Reboot Node 1.
- 2. On Node 2, uncluster SQL Server 7.0. Reboot Node 2.
- 3. On Node 1, install a clustered, named instance of SQL Server 2000 as a virtual server. This is not an upgrade process, but a side-by-side installation of SQL Server 7.0 and SQL Server 2000. Do not install the data to the same location/disk as Node 2. If you do, when you attempt to upgrade Node 2 from a SQL Server 7.0 to a SQL Server 2000 installation, Setup will fail.
- 4. On Node 1, run the Copy Database Wizard (CDW.exe) to move all databases and related information from the SQL Server 7.0 installation into the clustered, named instance of SQL Server 2000. For more information about the Copy Database Wizard, see <u>Using the Copy</u> Database Wizard or <u>How to upgrade databases online using the Copy</u> Database Wizard (Enterprise Manager).
- 5. On Node 1, uninstall SQL Server 7.0.
- 6. On Node 2, upgrade SQL Server 7.0 to SQL Server 2000 as the default instance.

You must install the binaries to a local drive and use a cluster disk for the data. This local drive is a path, which is a non-clustered disk valid on all nodes of the cluster. This drive on all nodes of the cluster must have at least 300 megabytes (MB) of available space.

7. On Node 2, upgrade the default instance of SQL Server 2000 to a clustered instance.

For more information, see <u>How to upgrade from a default instance to a</u> <u>default clustered instance of SQL Server 2000 (SQL Server Setup)</u>.

**Note** Optionally, you could create two named instances of SQL Server 2000 and use the Copy Database Wizard to upgrade both SQL Server 7.0 installations to a clustered, named instance of SQL Server 2000. This will provide better consistency, because all references to clustered installations of SQL Server 2000 will be in the form VirtualServer\Instance, rather than sometimes being just the servername, and sometimes both the servername and instancename.

# How to upgrade from a SQL Server 7.0 active/passive failover cluster (Setup)

#### To upgrade from a SQL Server 7.0 active/passive failover cluster

- 1. On Node 1, uncluster Microsoft® SQL Server<sup>™</sup> version 7.0. Reboot Node 1.
- 2. On Node 1, upgrade SQL Server 7.0 to SQL Server 2000 as the default instance.

You must install the binaries to a local drive and use a cluster disk for the data. This local drive is a path, which is a non-clustered disk valid on all nodes of the cluster. This drive on all nodes of the cluster must have at least 300 megabytes (MB) of available space.

3. On Node 1, upgrade the default instance of SQL Server 2000 to a clustered instance of SQL Server 2000.

For more information, see <u>How to upgrade from a default instance to a</u> <u>default clustered instance of SQL Server 2000 (SQL Server Setup)</u>.

# How to upgrade from a default instance to a default clustered instance of SQL Server 2000 (Setup)

**Note** This upgrade is from a default instance (a local installation where the data is on a local disk) to a clustered instance of Microsoft® SQL Server<sup>™</sup> 2000. Use this upgrade step if you want to have a default virtual server.

## To upgrade from a default instance to a default clustered instance of SQL Server 2000

- 1. On the **Welcome** screen of the **SQL Server Installation Wizard**, click **Next**.
- 2. On the **Computer Name** screen, click **Local Computer**. The computer you want to change from a default to a clustered instance should be displayed. You must be on the local computer to upgrade from a default to a clustered instance. Click **Next**.
- 3. On the **Installation Selection** screen, click **Upgrade, remove, or add components to an existing instance of SQL Server**. Click **Next**.
- 4. On the **Existing Installation** screen, click **Upgrade your existing installation to a clustered installation**. Click **Next**.
- 5. On the **Virtual Server Name** screen, enter a name for your virtual server. Click **Next**.
- 6. On the **Failover Clustering** screen, enter one IP address for each network configured for client access. That is, enter one IP address for each network on which the virtual server will be available to clients on a public (or mixed) network. Select the network for which you want to enter an IP address, and then enter the IP address. Click **Add**.

The IP address and the subnet are displayed. The subnet is supplied by Microsoft Cluster Service (MSCS). Continue to enter IP addresses for each installed network until you have populated all desired networks with an IP address. Click **Next**.

- 6. On the **Cluster Management** screen, review the failover cluster definition provided by SQL Server 2000. By default, all available nodes are selected. Remove any nodes that will not be part of the failover cluster definition for the virtual server you are creating. Click **Next**.
- 7. On the **Remote Information** screen, enter login credentials for the remote cluster node. The login credentials must have administrator privileges on the remote node(s) of the cluster. Click **Next**.
- 8. On the **Services Accounts** screen, select the service account(s) for the SQL Server services under which you want the failover cluster to run. Click **Next**.
- 9. On the **Setup Complete** screen, click **Finish**. If you need to restart the remote nodes in the failover cluster, you will be instructed to do so in the **Setup Complete** screen.

# How to upgrade from a local default instance to a clustered, named instance of SQL Server 2000 (Setup)

To upgrade from a local default instance to a named clustered instance of SQL Server 2000

- 1. Install a clustered, named instance of Microsoft® SQL Server<sup>™</sup> 2000.
- 2. Run the Copy Database Wizard (CDW.exe) to move all databases and related information into the clustered, named instance of SQL Server 2000. For more information about the Copy Database Wizard, see Using the Copy Database Wizard or How to upgrade databases online using the Copy Database Wizard (Enterprise Manager).
- 3. Optionally, you can uninstall the default instance of SQL Server 2000.

## **Collation Options for International Support**

In Microsoft® SQL Server<sup>™</sup> 2000, it is not required to separately specify code page and sort order for character data, and the collation used for Unicode data. Instead, specify the collation name and sorting rules to use. The term, collation, refers to a set of rules that determine how data is sorted and compared. Character data is sorted using rules that define the correct character sequence, with options for specifying case-sensitivity, accent marks, kana character types, and character width. Microsoft SQL Server 2000 collations include these groupings:

• Windows collations

Windows collations define rules for storing character data based on the rules defined for an associated Windows locale. The base Windows collation rules specify which alphabet or language is used when dictionary sorting is applied, as well as the code page used to store non-Unicode character data. For more information, see <u>Collations</u>.

• SQL collations

SQL collations are provided for compatibility with sort orders in earlier versions of Microsoft SQL Server. For more information, see <u>Using</u> <u>SQL Collations</u>.

#### **Changing Collations After Setup**

When you set up SQL Server 2000, it is important to use the correct collation settings. You can change collation settings after running Setup, but you must rebuild the databases and reload the data. It is recommended that you develop a standard within your organization for these options. Many server-to-server activities can fail if the collation settings are not consistent across servers.

#### See Also

<u>Collation Settings in Setup</u> How to rebuild the **master** database (Rebuild Master utility) Selecting a SQL Collation Windows Collation Designators

## **Collation Settings in Setup**

Use the **Collation Settings** screen to modify default collation settings. Use the **Windows Locale** option to match collation settings in instances of Microsoft® SQL Server<sup>™</sup> 2000. Use **SQL Collations** to match settings that are compatible with the sort orders in earlier versions of SQL Server.

#### Windows Locale

Change the default settings for **Windows Locale** (Windows collation) only if your installation of SQL Server must match the collation settings used by another instance of SQL Server 2000, or must match the Windows locale of another computer.

#### **Collation Designator**

Select the name of a specific Windows collation from the list, for example:

- Use Latin1\_General for the U.S. English character set (code page 1252).
- Use **Modern\_Spanish** for all variations of Spanish, which also use the same character set as U.S. English (code page 1252).
- Use **Arabic** for all variations of Arabic, which use the Arabic character set (code page 1256).
- Use **Japanese\_Unicode** for the Unicode version of Japanese (code page 932), which has a different sort order from **Japanese**, but the same code page (932).

For more information, see <u>Windows Collation Designators</u>.

#### Sort Order

Select Sort Order options to use with the Collation Designator selected. Binary is the fastest sorting order, and is case-sensitive. If **Binary** is selected, the **Case-sensitive**, **Accent-sensitive**, **Kana-sensitive**, and **Width-sensitive** options are not available. For more information, see <u>Windows Collation Sorting Styles</u>.

### **SQL Collations**

The **SQL Collations** option is used for compatibility with earlier versions of Microsoft SQL Server. Select this option to match settings compatible with SQL Server version 7.0, SQL Server version 6.5, or earlier. For more information, see <u>SQL Collations</u>.

## **Windows Collation Sorting Styles**

On the **Collation Settings** screen you can choose **Binary** sort order, or you can define the sorting styles to use with the Collation Designator (Windows collation name) selected.

**Note** For Windows collations, the **nchar**, **nvarchar**, and **ntext** data types have the same sorting behavior as **char**, **varchar**, and **text** data types. For more information, see <u>SQL Server Collation Fundamentals</u>.

Sort order	Description
Binary	Sorts and compares data in Microsoft® SQL Server <sup>™</sup> tables based on the bit patterns defined for each character. Binary sort order is case-sensitive, that is lowercase precedes uppercase, and accent-sensitive. This is the fastest sorting order. If this option is not selected, SQL Server follows sorting
	and comparison rules as defined in dictionaries for the associated language or alphabet.
Case-sensitive	Specifies that SQL Server distinguish between uppercase and lowercase letters.
	If not selected, SQL Server considers the uppercase and lowercase versions of letters to be equal. SQL Server does not define whether lowercase letters sort lower or higher in relation to uppercase letters when Case- sensitive is not selected.
Accent-sensitive	Specifies that SQL Server distinguish between accented and unaccented characters. For example, 'a' is not equal to 'á'.
	If not selected, SQL Server considers the accented and unaccented versions of letters to be equal.
Kana-sensitive	Specifies that SQL Server distinguish between the two types of Japanese kana characters: Hiragana and

	Katakana. If not selected, SQL Server considers Hiragana and Katakana characters to be equal.
Width-sensitive	Specifies that SQL Server distinguish between a single- byte character (half-width) and the same character when represented as a double-byte character (full-width). If not selected, SQL Server considers the single-byte and double-byte representation of the same character to be equal.

### See Also

Collation Settings in Setup

Windows Collation Designators

## **Windows Collation Designators**

Use this table to synchronize collation settings with another Windows locale.

In Control Panel, find the Windows locale name in the Regional Settings application (Microsoft® Windows NT® 4.0, Microsoft Windows® 98, and Microsoft Windows 95) or the Regional Options application (Microsoft Windows 2000), and then use this table to find the corresponding Collation Designator and code page.

	LCID		Code
Windows locale	(locale ID)	<b>Collation designator</b>	page
Afrikaans	0xx436	Latin1_General	1252
Albanian	0x41C	Albanian	1250
Arabic (Saudi Arabia)	0x401	Arabic	1256
Arabic (Iraq)	0x801	Arabic	1256
Arabic (Egypt)	0xC01	Arabic	1256
Arabic (Libya)	0x1001	Arabic	1256
Arabic (Algeria)	0x1401	Arabic	1256
Arabic (Morocco)	0x1801	Arabic	1256
Arabic (Tunisia)	0x1C01	Arabic	1256
Arabic (Oman)	0x2001	Arabic	1256
Arabic (Yemen)	0x2401	Arabic	1256
Arabic (Syria)	0x2801	Arabic	1256
Arabic (Jordan)	0x2C01	Arabic	1256
Arabic (Lebanon)	0x3001	Arabic	1256
Arabic (Kuwait)	0x3401	Arabic	1256
Arabic (United Arab Emirates)	0x3801	Arabic	1256
Arabic (Bahrain)	0x3C01	Arabic	1256
Arabic (Qatar)	0x4001	Arabic	1256
Basque	0x42D	Latin1_General	1252
Byelorussian	0x423	Cyrillic_General	1251
Bulgarian	0x402	Cyrillic_General	1251

Catalan	0x403	Latin1_General	1252
Chinese (Taiwan)	0x30404	Chinese_Taiwan_Bopomofo	950
Chinese (Taiwan)	0x404	Chinese_Taiwan_Stroke	950
Chinese (People's Republic of China)	0x804	Chinese_PRC	936
Chinese (People's Republic of China)	0x20804	Chinese_PRC_Stroke	936
Chinese (Singapore)	0x1004	Chinese_PRC	936
Croatia	0x41a	Croatian	1250
Czech	0x405	Czech	1250
Danish	0x406	Danish_Norwegian	1252
Dutch (Standard)	0x413	Latin1_General	1252
Dutch (Belgium)	0x813	Latin1_General	1252
English (United States)	0x409	Latin1_General	1252
English (Britain)	0x809	Latin1_General	1252
English (Canada)	0x1009	Latin1_General	1252
English (New Zealand)	0x1409	Latin1_General	1252
English (Australia)	0xC09	Latin1_General	1252
English (Ireland)	0x1809	Latin1_General	1252
English (South Africa)	0x1C09	Latin1_General	1252
English (Carribean)	0x2409	Latin1_General	1252
English (Jamaican)	0x2009	Latin1_General	1252
Estonian	0x425	Estonian	1257
Faeroese	0x0438	Latin1_General	1252
Farsi	0x429	Arabic	1256
Finnish	0x40B	Finnish_Swedish	1252
French (Standard)	0x40C	French	1252
French (Belgium)	0x80C	French	1252
French (Switzerland)	0x100C	French	1252
French (Canada)	0xC0C	French	1252
French (Luxembourg)	0x140C	French	1252
Georgian (Modern	0x10437	Georgian_Modern_Sort	1252

Sort)			
German (PhoneBook	0x10407	German_PhoneBook	1252
Sort)			
German (Standard)	0x407	Latin1_General	1252
German (Switzerland)	0x807	Latin1_General	1252
German (Austria)	0xC07	Latin1_General	1252
German (Luxembourg)	0x1007	Latin1_General	1252
German	0x1407	Latin1_General	1252
(Liechtenstein)			
Greek	0x408	Greek	1253
Hebrew	0x40D	Hebrew	1255
Hindi	0x439	Hindi	Unicode
			only
Hungarian	0x40E	Hungarian	1250
Hungarian	0x104E	Hungarian_Technical	1250
Icelandic	0x40F	Icelandic	1252
Indonesian	0x421	Latin1_General	1252
Italian	0x410	Latin1_General	1252
Italian (Switzerland)	0x810	Latin1_General	1252
Japanese	0x411	Japanese	932
Japanese (Unicode)	0x10411	Japanese_Unicode	932
Korean (Extended Wansung)	0x412	Korean_Wansung	949
Korean	0x412	Korean_Wansung_Unicode	949
Latvian	0x426	Latvian	1257
Lithuanian	0x427	Lithuanian	1257
Lithuanian	0x827	Lithuanian_Classic	1257
Macedonian	0x41C	Cyrillic_General	1251
Norwegian (Bokmål)	0x414	Danish_Norwegian	1252
Norwegian (Nynorsk)	0x814	Danish_Norwegian	1252
Polish	0x415	Polish	1250
Portuguese (Standard)	0x816	Latin1_General	1252
Portuguese (Brazil)	0x416	Latin1_General	1252

Romanian	0x418	Romanian	1250
Russian	0x419	Cyrillic_General	1251
Serbian (Latin)	0x81A	Cyrillic_General	1251
Serbian (Cyrillic)	0xC1A	Cyrillic_General	1251
Slovak	0x41B	Slovak	1250
Slovenian	0x424	Slovenian	1250
Spanish (Mexico)	0x80A	Traditional_Spanish	1252
Spanish (Traditional Sort)	0x40A	Traditional_Spanish	1252
Spanish (Modern Sort)	0xC0A	Modern_Spanish	1252
Spanish (Guatemala)	0x100A	Modern_Spanish	1252
Spanish (Costa Rica)	0x140A	Modern_Spanish	1252
Spanish (Panama)	0x180A	Modern_Spanish	1252
Spanish (Dominican Republic)	0x1C0A	Modern_Spanish	1252
Spanish (Venezuela)	0x200A	Modern_Spanish	1252
Spanish (Colombia)	0x240A	Modern_Spanish	1252
Spanish (Peru)	0x280A	Modern_Spanish	1252
Spanish (Argentina)	0x2C0A	Modern_Spanish	1252
Spanish (Ecuador)	0x300A	Modern_Spanish	1252
Spanish (Chile)	0x340A	Modern_Spanish	1252
Spanish (Uruguay)	0x380A	Modern_Spanish	1252
Spanish (Paraguay)	0x3C0A	Modern_Spanish	1252
Spanish (Bolivia)	0x400A	Modern_Spanish	1252
Swedish	0x41D	Finnish_Swedish	1252
Thai	0x41E	Thai	874
Turkish	0x41F	Turkish	1254
Ukrainian	0x422	Ukrainian	1251
Urdu	0x420	Arabic	1256
Vietnamese	0x42A	Vietnamese	1258

Collation Settings in Setup Collations Windows Collation Sorting Styles Windows Collation Name

## **Using SQL Collations**

SQL collation settings correspond to the type of installation. In general, choose a SQL collation that supports the Windows locale most commonly used at your site. For more information about identifying your site Windows Locale, see Regional Settings in Windows Control Panel. In many cases, a computer will run the Windows locale that matches the language requirements of the user, so Setup automatically detects the Windows locale and chooses the appropriate SQL collation.

SQL collations control:

- The code page used for storing non-Unicode data in Microsoft® SQL Server<sup>™</sup>.
- The rules governing how SQL Server sorts and compares characters stored in both Unicode and non-Unicode data types.

Choose a SQL collation if:

- You use the replication feature with existing instances of SQL Server version 6.5 or SQL Server version 7.0
- Your application code depends on the behaviors of the previous SQL Server collations.

An upgrade of SQL Server 7.0 to SQL Server 2000 keeps the previous SQL collation settings; no collation choice is required.

Use this table to determine if you need to make a collation choice, and if so, which collation you should choose.

Installation you want	Collation to choose
To install on a new system with no	Use the locale identified by Setup,
compatibility requirements for	and then choose the desired binary,
synchronizing with any type of	case, or other options.
existing system	

	For this release of SQL Server, when Setup detects that the computer is running the U.S. English locale, Setup automatically selects the SQL collation: <b>Dictionary order, case-</b> <b>insensitive, for use with 1252</b> <b>character set.</b>
	To select the equivalent Windows collation, select <b>Collation</b> <b>designator</b> , choose the <b>Latin1_General</b> collation designator, do not select <b>case-sensitive</b> , and select <b>accent-sensitive</b> .
To upgrade an installation of SQL Server 6.5 or SQL Server 7.0 to a default instance of SQL Server 2000, or to install a default instance of SQL Server 2000 that will version switch with an installation of SQL Server 6.5	Use the SQL collation chosen by Setup.
To synchronize (for example, to replicate) with an existing instance of SQL Server 2000	Select SERVERPROPERTY(N'Collation') on the existing instance, and specify that collation. If the collation name of the existing instance starts with SQL, select the same SQL collation in Setup. If the collation name of the existing instance does not start with SQL, the collation name refers to a Windows collation name and consists of the collation designator name followed by a description of what binary, case, accent, kana and width sensitivity options are specified. Select the same Windows collation designator and sorting options in

	Setup.
To synchronize with an existing installation of SQL Server 6.5 or SQL Server 7.0	<ul> <li>Execute sp_helpsort on the existing system, and then use the sort ID to select a SQL collation to make your instance of SQL Server 2000 compatible with an existing installation.</li> <li>For more information, see <u>Selecting a SQL Collation</u>.</li> </ul>
To synchronize with a Windows locale of another computer	<ul> <li>In Control Panel, find the locale name from the Regional Settings application (Microsoft Windows NT® 4.0, Microsoft Windows® 98, and Microsoft Windows 95), or from the Regional Options application (Microsoft Windows 2000), and then use the table provided in the topic Windows Collation Designators. Set the sorting options, as explained in the topic Windows Collation Sorting Styles.</li> </ul>

**Note** When you perform an action that depends on collations, the SQL Server collation used by the referenced object must use a code page supported by the operating system running on the computer. For more information, see <u>Specifying</u> <u>Collations</u>.

#### See Also

Examples of SQL Collations

**Selecting Collations** 

<u>sp\_helpsort</u>

Setting Client Code Pages
SQL Server Collation Fundamentals
SERVERPROPERTY

## **Examples of SQL Collations**

These are examples of SQL collations listed on the **Collation Settings** screen in Microsoft<sup>®</sup> SQL Server<sup>™</sup> 2000 Setup:

#### Binary order, for use with the 437 (U.S. English) character set.

This collation uses binary sort order (simple sorting based on coded value) with the U.S. English character set (code page 437 - MS-DOS Latin US).

In Transact-SQL, the string SQL\_Latin1\_General\_Cp437\_BIN is used to designate this setting.

## Dictionary order, case-insensitive, accent-insensitive, for use with 1252 character set.

This collation uses the dictionary sorting rules for the U.S. English character set (code page 1252 - Windows Latin 1 ANSI, sort order ID 54). Uppercase or lowercase characters and accent marks are not considered when sorting.

In Transact-SQL, the string SQL\_Latin1\_General\_CP1\_CI\_AI is used to designate this setting.

## Romanian dictionary order, case-sensitive, for use with the 1250 (Central European) character set.

This collation uses the dictionary order sorting rules for the Romanian language, and uses the Central European character set (code page 1250, sort order ID 89).

In Transact-SQL, the string SQL\_Romanian\_Cp1250\_CS\_AS is used to designate this setting.

#### See Also

Collation Settings in Setup Selecting a SQL Collation Using SQL Collations SQL Collation Name

## **Selecting a SQL Collation**

When selecting a SQL collation in the **Collations Settings** screen, use the following table to make the installation of Microsoft® SQL Server<sup>™</sup> 2000 compatible with an installation of an earlier version of SQL Server.

**Note** Each SQL Collation name has an equivalent string in T-SQL code. For a list of sort order identifiers and the T-SQL version, see <u>SQL Collation Name</u>.

In the table, the left column lists the sort order ID of an instance of SQL Server 7.0 or SQL Server 6.5. The right column lists the SQL Server 2000 collation recommended for compatibility.

Sort order ID	SQL collation name
30	Binary order, for use with the 437 (U.S. English) character set.
31	Dictionary order, case-sensitive, for use with the 437 (U.S. English) character set.
32	Dictionary order, case-insensitive, for use with the 437 (U.S. English) character set.
33	Dictionary order, case-insensitive, uppercase preference, for use with the 437 (U.S. English) character set.
34	Dictionary order, case-insensitive, accent-insensitive, for use with the 437 (U.S. English) character set.
40	Binary order, for use with the 850 (Multilingual) character set.
41	Dictionary order, case-sensitive, for use with the 850 (Multilingual) character set.
42	Dictionary order, case-insensitive, for use with the 850 (Multilingual) character set.
43	Dictionary order, case-insensitive, uppercase preference, for use with the 850 (Multilingual) character set.
44	Dictionary order, case-insensitive, accent-insensitive, for use with the 850 (Multilingual) character set.
49	Strict compatibility with version 1. <i>x</i> case-insensitive databases, for use with the 850 (Multilingual) character set.

50	Binary order for use with 1252 character set.
51	Dictionary order, case-sensitive, for use with 1252 character set.
52	Dictionary order, case-insensitive, for use with 1252 character set.
53	Dictionary order, case-insensitive, uppercase preference, for use with 1252 character set.
54	Dictionary order, case-insensitive, accent-insensitive, for use with 1252 character set.
55	Alternate dictionary order, case-sensitive, for use with the 850 (Multilingual) character set.
56	Alternate dictionary order, case-insensitive, uppercase preference, for use with the 850 (Multilingual) character set.
57	Alternate dictionary order, case-insensitive, accent-insensitive, for use with the 850 (Multilingual) character set.
58	Scandinavian dictionary order, case-insensitive, uppercase preference, for use with the 850 (Multilingual) character set.
59	Scandinavian dictionary order, case-sensitive, for use with the 850 (Multilingual) character set.
60	Scandinavian dictionary order, case-insensitive, for use with the 850 (Multilingual) character set.
61	Alternate dictionary order, case-insensitive, for use with the 850 (Multilingual) character set.
71	Latin-1 case-sensitive, for use with 1252 character set.
72	Latin-1 case-insensitive, for use with 1252 character set.
73	Danish/Norwegian case-sensitive sort order for code page 1252.
74	Finnish/Swedish case-sensitive sort order for code page 1252.
75	Icelandic case-sensitive sort order for code page 1252.
80	Binary order, for use with the 1250 (Central European) character set.
81	Dictionary order, case-sensitive, for use with the 1250 (Central European) character set.
82	Dictionary order, case-insensitive, for use with the 1250

	(Central European) character set.
83	Czech dictionary order, case-sensitive, for use with the 1250 (Central European) character set.
84	Czech dictionary order, case-insensitive, for use with the 1250 (Central European) character set.
85	Hungarian dictionary order, case-sensitive, for use with the 1250 (Central European) character set.
86	Hungarian dictionary order, case-insensitive, for use with the 1250 (Central European) character set.
87	Polish dictionary order, case-sensitive, for use with the 1250 (Central European) character set.
88	Polish dictionary order, case-insensitive, for use with the 1250 (Central European) character set.
89	Romanian dictionary order, case-sensitive, for use with the 1250 (Central European) character set.
90	Romanian dictionary order, case-insensitive, for use with the 1250 (Central European) character set.
91	Croatian dictionary order, case-sensitive, for use with the 1250 (Central European) character set.
92	Croatian dictionary order, case-insensitive, for use with the 1250 (Central European) character set.
93	Slovak dictionary order, case-sensitive, for use with the 1250 (Central European) character set.
94	Slovak dictionary order, case-insensitive, for use with the 1250 (Central European) character set.
95	Slovenian dictionary order, case-sensitive, for use with the 1250 (Central European) character set.
96	Slovenian dictionary order, case-insensitive, for use with the 1250 (Central European) character set.
97	Windows Polish case-sensitive sort order for code page 1250.
98	Windows Polish case-insensitive sort order for code page 1250.
104	Binary order, for use with the 1251 (Cyrillic) character set.
105	Dictionary order, case-sensitive, for use with the 1251 (Cyrillic) character set.

106	Dictionary order, case-insensitive, for use with the 1251 (Cyrillic) character set.
107	Ukrainian dictionary order, case-sensitive, for use with the 1251 (Cyrillic) character set.
108	Ukrainian dictionary order, case-insensitive, for use with the 1251 (Cyrillic) character set.
112	Binary order, for use with the 1253 (Greek) character set.
113	Dictionary order, case-sensitive, for use with the 1253 (Greek) character set.
114	Dictionary order, case-insensitive, for use with the 1253 (Greek) character set.
120	Mixed dictionary order, for use with the 1253 (Greek) character set.
121	Dictionary order, case-sensitive, accent-sensitive, for use with the 1253 (Greek) character set.
124	Dictionary order, case-insensitive, accent-insensitive, for use with the 1253 (Greek) character set.
128	Binary order, for use with the 1254 (Turkish) character set.
129	Dictionary order, case-sensitive, for use with the 1254 (Turkish) character set.
130	Dictionary order, case-insensitive, for use with the 1254 (Turkish) character set.
136	Binary order, for use with the 1255 (Hebrew) character set.
137	Dictionary order, case-sensitive, for use with the 1255 (Hebrew) character set.
138	Dictionary order, case-insensitive, for use with the 1255 (Hebrew) character set.
144	Binary order, for use with the 1256 (Arabic) character set.
145	Dictionary order, case-sensitive, for use with the 1256 (Arabic) character set.
146	Dictionary order, case-insensitive, for use with the 1256 (Arabic) character set.
152	Binary order, for use with the 1257 (Baltic) character set.
153	Dictionary order, case-sensitive, for use with the 1257 (Baltic) character set.

Estonian dictionary order, case-sensitive, for use with the
1257 (Baltic) character set.
Estonian dictionary order, case-insensitive, for use with the 1257 (Baltic) character set.
Latvian dictionary order, case-sensitive, for use with the 1257 (Baltic) character set.
Latvian dictionary order, case-insensitive, for use with the 1257 (Baltic) character set.
Lithuanian dictionary order, case-sensitive, for use with the 1257 (Baltic) character set.
Lithuanian dictionary order, case-insensitive, for use with the 1257 (Baltic) character set.
Danish/Norwegian dictionary order, case-insensitive, uppercase preference, for use with 1252 character set.
Swedish/Finnish (Standard) dictionary order, case-insensitive, uppercase preference, for use with 1252 character set.
Swedish/Finnish (Phone) dictionary order, case-insensitive, uppercase preference, for use with 1252 character set.
Icelandic dictionary order, case-insensitive, uppercase preference, for use with 1252 character set.
Binary order, for use with the 932 (Japanese) character set.
Dictionary order, case-insensitive, for use with the 932 (Japanese) character set
Binary order, for use with the 949 (Korean) character set.
Dictionary order, case-insensitive, for use with the 949 (Korean) character set.
Binary order, for use with the 950 (Traditional Chinese) character set.
Dictionary order, case-insensitive, for use with the 950 (Traditional Chinese) character set.
Binary order, for use with the 936 (Simplified Chinese) character set.

199	Dictionary order, case-insensitive, for use with the 936
	(Simplified Chinese) character set.
200	Dictionary order, case-sensitive, for use with the 932
	(Japanese) character set.
201	Dictionary order, case-sensitive, for use with the 949 (Korean)
	character set.
202	Dictionary order, case-sensitive, for use with the 950
	(Traditional Chinese) character set.
203	Dictionary order, case-sensitive, for use with the 936
	(Simplified Chinese) character set.
204	Binary order, for use with the 874 (Thai) character set.
205	Dictionary order, case-insensitive, for use with the 874 (Thai)
	character set.
206	Dictionary order, case-sensitive, for use with the 874 (Thai)
	character set.

#### See Also

Examples of SQL Collations

Collation Settings in Setup

**Specifying Collations** 

SQL Collations

## **Setting Client Code Pages**

The code pages a client uses are determined by your operating system settings.

To set client code pages in the Windows NT, Windows 2000, or Windows 98 operating systems

## **Upgrading Character Set, Sort Order, and Collation**

Microsoft® SQL Server<sup>™</sup> 2000 supports several different ways to specify collations. You no longer have to separately specify the code page used for character data, the sort order used for character data, and the collation used for Unicode data. When you upgrade, SQL collations can be specified for compatibility with existing instances of SQL Server.

Because the default collation for an instance of Microsoft SQL Server is defined during setup, it is important to become familiar with collation settings in SQL Server 2000 when:

- Your application code depends in some way on the behavior of previous SQL Server collations.
- You are going to use the replication feature with existing installations of SQL Server 6.5 or SQL Server 7.0.
- You must store character data that reflects multiple languages.

#### See Also

**Collation Options for International Support** 

**Collations** 

**Selecting Collations** 

Specifying the Default Collation for an Instance of SQL Server

# **Changing Collation Settings After Installing**

Collation settings, which include character set, sort order, and other localespecific settings, are fundamental to the structure of all Microsoft® SQL Server<sup>TM</sup> 2000 databases. To change one or more of these settings, you must rebuild the **master** and user databases.

#### See Also

**Collation Settings in Setup** 

**Collations** 

How to rebuild the master database (Rebuild Master utility)

## After Installing or Upgrading to SQL Server 2000

For a standard installation, components include:

- The SQL Server relational database engine.
- System databases used to store system level information such as login and configuration settings and for use as database templates.
- The **pubs** and **Northwind** sample databases, provided as learning tools.
- Stored procedures, a recompiled collection of Transact-SQL statements.
- Interactive management tools used for administering SQL Server.
- SQL Server Books Online, the complete documentation for SQL Server 2000.

#### See Also

Books Online Management Tools Select Components SQL Stored Procedures Using the Start Menu System and Sample Databases System Databases and Data

# **Using the Start Menu**

Microsoft® SQL Server<sup>™</sup> 2000 Setup creates the **Microsoft SQL Server** program group on the **Start** menu in the **Programs** group. From the **Microsoft SQL Server** program group, you can start:

- Books Online.
- Client Network Utility.
- Configure SQL XML Support in IIS.
- Enterprise Manager.
- Import and Export Data.
- Profiler.
- Query Analyzer.
- Server Network Utility.
- Service Manager.

#### **Microsoft SQL Server-Switch Program Group**

If you install an instance of Microsoft SQL Server 2000 (default or named) on the same computer with an installation of Microsoft SQL Server version 6.5, Setup removes the **Microsoft SQL Server 6.5** program group and adds the **Microsoft SQL Server-Switch** program group. SQL Server 6.5 and SQL Server 2000 cannot run at the same time, but using the **Server-Switch** program group, you can switch between the two versions.

On the **Start** menu, only the program group of the active version of SQL Server is accessible at any given time. The nonactive version is shown in the **Server-Switch** program group so you can quickly switch from one version to another.

The **Microsoft SQL Server-Switch** program group contains these options:

- Microsoft SQL Server 6.5 or Microsoft SQL Server 2000 (the nonactive version)
- SQL Server Upgrade Wizard
- Uninstall SQL Server 6.5

#### See Also

Switching Between SQL Server 6.5 and SQL Server 2000

# **System and Sample Databases**

When Microsoft® SQL Server<sup>™</sup> 2000 is installed, Setup creates the database and log files shown in this table.

Database	Database file	Log file
master	Master.mdf	Mastlog.ldf
model	Model.mdf	Modellog.ldf
msdb	Msdbdata.mdf	Msdblog.ldf
tempdb	Tempdb.mdf	Templog.ldf
pubs	Pubs.mdf	Pubs_log.ldf
Northwind	Northwnd.mdf	Northwnd.ldf

The system databases are **master**, **model**, **msdb**, and **tempdb**. The sample databases, **pubs** and **Northwind**, are provided as learning tools. (Names of these databases are case-sensitive.) Many of the examples in SQL Server Books Online are based on the sample databases.

**Note** The default location of the database and log files is Program Files\Microsoft SQL Server\Mssql\Data. This location may vary if the default location was changed when SQL Server was installed.

#### See Also

Northwind Sample Database

pubs Sample Database

System Databases and Data

# **Locating Directories and Files**

The following tables and illustration show the default locations of directories and files for Microsoft® SQL Server<sup>™</sup> 2000 (primarily for the default instance). Depending on the options you install, all of the files listed in the tables may not appear on your computer, and others not listed may be included.

Paths listed here are default paths, and may vary if locations were changed during installation. Both program and data file locations can be changed, but the location of shared tools cannot be changed.

**IMPORTANT** Do not delete any of the following directories or their contents: Binn, Data, Ftdata, HTML, or 1033. You may delete other directories, if necessary; however, you may not be able to retrieve any lost functionality or data without uninstalling and reinstalling SQL Server 2000.

Do not delete or modify any of the .htm files in the HTML directory. They are required for SQL Server Enterprise Manager and other tools to function properly.

## Shared Files for All Instances of SQL Server 2000

This table shows the locations for the shared files for both default and named instances of SQL Server 2000.

Location	Description
\Program Files\Microsoft SQL	Dynamic-link libraries (DLLs) for
Server\80\Com	Component Object Model (COM)
	objects.
\Program Files\Microsoft SQL	Resource files (RLLs) used by the
Server\80\Com\Binn\Resources\1033	DLLs in this COM directory. (Note:
	1033 is for U.S. English; localized
	versions use different directory
	numbers.)
\Program Files\Microsoft SQL	Microsoft Windows NT® client
Server\80\Tools\Binn	executable files.
\Program Files\Microsoft SQL	Resource files used by the DLLs in

Server\80\Tools\Binn\Resources\1033	the Tools\Binn directory.
\Program Files\Microsoft SQL	SQL Server Books Online files,
Server\80\Tools\Books	including online Help files.
Program Files\Microsoft SQL	Header files, library files, and
Server\80\Tools\DevTools\	sample programs for use by
	developers.
Program Files\Microsoft SQL	Microsoft Management Console
Server\80\Tools\Html	(MMC) and SQL Server HTML
	files.
Program Files\Microsoft SQL	Boilerplate files with SQL scripts to
Server\80\Tools\Templates	help you create objects in the
	database.

## **Program and Data Files for the Default Instance of SQL Server** 2000

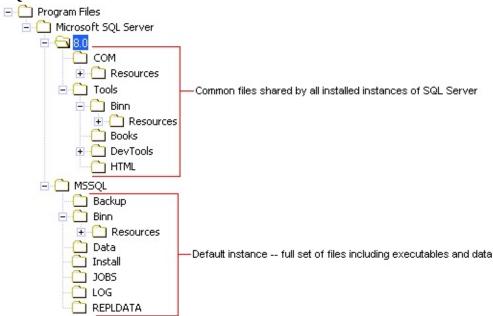
This table shows the locations of the program and data files for the default instance of SQL Server 2000. These are the default file locations, which can be changed during installation.

Location	Description
\Program Files\Microsoft SQL	Default location for backup files.
Server\Mssql\Backup	
\Program Files\Microsoft SQL	Microsoft Windows NT server
Server\Mssql\Binn	executable files and DLL files for
	extended stored procedures.
\Program Files\Microsoft SQL	Resource files used by the DLLs in this
Server\Mssql\Binn\Resources\1033	Binn directory.
Program Files\Microsoft SQL	System and sample database files.
Server\Mssql\Data	
Program Files\Microsoft SQL	Full-text catalog files.
Server\Mssql\Ftdata	
Program Files\Microsoft SQL	Scripts run during Setup and resulting
Server\Mssql\Install	output files.
Program Files\Microsoft SQL	Storage location for temporary job
Server\Mssql\Jobs	output files.

Program Files\Microsoft SQL Server\Mssql\Log	Error log files.
Program Files\Microsoft SQL Server\Mssql\Repldata	Working directory for replication tasks.
Program Files\Microsoft SQL Server\Mssql\Upgrade	Files used for version upgrade from SQL Server version 6.5 to SQL Server 2000.

## File Locations for the Default Instance of SQL Server 2000

This illustration shows the file locations for the default instance of Microsoft® SQL Server<sup>TM</sup> 2000.



#### See Also

File Paths for SQL Server 2000

# **Changing Passwords and User Accounts**

Microsoft® SQL Server<sup>™</sup> 2000 services accounts and passwords are linked to Microsoft Windows® user accounts and passwords. Changes in one location may require changes in the other.

## **Changing SQL Server Services Accounts After Install**

After you have installed SQL Server 2000, use SQL Server Enterprise Manager to change the assigned password or other properties of any SQL Server—related service. Each service must be changed individually. The new user account takes effect when the service is restarted. You should not change the passwords for any of the SQL Server service accounts when a failover cluster node is down or offline. If you have to do this, you will need to reset the password again using Enterprise Manager when all nodes are back online.

If you are running Microsoft Windows NT®, and you select to change the current service account for SQL Server to a non-administrator account (and the current service account for SQL Server is not an administrator account), the Valid Administrator Login dialog box is displayed. SQL Server must have administrator privileges to change security entries, so you must enter the user name, password, and domain to impersonate the non-administrator service account you have selected.

Once you have specified this information, all objects are granted full control permission. The location of the objects is determined by the following:

- Permissions are set for all files in the binary and data installation locations for the specific instances.
- Registry permissions depend on whether the instance is default or named:

For a default instance, permissions are applied only to the entries listed below the HKLM\Software\Microsoft\MSSQLServer entry:

• SQLServerAgent

- Replication
- Providers
- Setup
- Tracking
- MSSQLServer

For a named instance, permissions are applied to the entire HKLM\Software\Microsoft\MicrosoftSQLServer\80 entry.

The following rights are granted to the accounts:

- SeServiceLogonRight, which allows the account to run as a service.
- SeLockMemoryPrivilege, which allows the account to use the AWE memory feature of SQL Server.
- SeTcbPrivilege, which allows the account to impersonate other accounts.

If you are running SQL Server in a failover cluster configuration, permissions are also set for all files in the binary and data installation locations for all nodes in the cluster. Permission is also granted for the service account on the Cluster Object.

**Note** If you are running Microsoft Windows 2000 and want to use the Windows 2000 Encrypted File System to encrypt any SQL Server files, you must unencrypt the files before you can change the SQL Server service accounts. If you do not unencrypt the files and then reset the SQL Server service accounts, you cannot unencrypt the files.

You can change the SQLServerAgent service account to a non Microsoft Windows NT® 4.0 administrator account. However, the Windows NT 4.0

account must be a member of the **sysadmin** fixed server role to run SQL Server Agent.

To change the MSSQLServer services login (Enterprise Manager)

# **Renaming a Server**

When you change the name of the computer that is running Microsoft® SQL Server<sup>TM</sup> 2000, the new name is recognized during SQL Server startup. You do not have to run Setup again to reset the computer name.

You can connect to SQL Server using the new computer name after you have restarted the server. However, to correct the **sysservers** system table, you should manually run these procedures:

```
sp_dropserver <old_name>
go
sp_addserver <new_name>
go
```

## **Issues with Remote Logins and Replication**

If the computer has any remote logins, for example, if it is a replication Publisher or Distributor, **sp\_dropserver** may generate an error similar to this:

Server: Msg 15190, Level 16, State 1, Procedure sp\_dropserver, Line 4 There are still remote logins for the server 'SERVER1'.

To resolve the error, you may need to drop remote logins for this server. If replication is installed, disable replication on the server before running the **sp\_dropserver** stored procedure.

#### To disable replication using the SQL Server Enterprise Manager

- 1. Expand a server group, and then expand the Distributor (the server that contains the distribution database).
- 2. Right-click the Replication folder, and then click **Disable Publishing**.
- 3. Complete the steps in the Disable Publishing and Distribution Wizard.

## **Deploying SQL Server After Initial Installation**

Microsoft® SQL Server<sup>™</sup> 2000 includes a new method for distributing a disk image of an installation. When an installation is first created, it is marked as a new installation. When the server is restarted after installation, SQL Server 2000 verifies that the server name has not changed. If the server name has changed, an automatic correction is made.

This functionality allows Independent Service Vendors to install SQL Server 2000, stop the server, clone the disk image, and then distribute it as required. On the first startup of the distributed server, the name correction is made.

This process can be done only one time. If the server is restarted and then stopped, a new SQL Server installation must be created to be distributed as an image during deployment.

# **Installing Full-Text Search and Indexing Tools**

The full-text search engine (Microsoft Search service) is installed by default with a typical installation of Microsoft® SQL Server<sup>™</sup> 2000, Standard and Enterprise editions.

**IMPORTANT** If upgrading from SQL Server 7.0 to SQL Server 2000 and full-text search is not installed in SQL Server 7.0, install full-text search as an additional component after the upgrade is completed.

#### **Microsoft Indexing Service Version 2.0**

In addition to using full-text search on character columns in SQL Server data, you can use Microsoft Indexing Service along with Microsoft Search service to make textual queries against data residing in the file system. This indexing service is included in Microsoft Windows® 2000. Microsoft Windows NT® users can install the indexing service from the Microsoft Windows NT 4.0 Option Pack.

#### To install Microsoft Indexing Service 2.0 (Windows NT only)

- 1. Install Windows NT 4.0 Option Pack.
- 2. Select **Index Server 2.0**. An error appears stating that Index Server 2.0 did not install properly.
- 3. Install Windows NT 4.0 Service Pack 4.
- 4. Reinstall Windows NT 4.0 Option Pack. You are not required to change any settings.
- 5. Optional step: Repeat the reinstallation of Windows NT 4.0 Service Pack 4.

Repeating this procedure ensures proper installation.

## See Also

Full-text Querying of File Data

How to add components to an instance of SQL Server 2000 (Setup)

# **Configuring SQL Server 2000 After Upgrading**

After the server is upgraded to Microsoft® SQL Server<sup>™</sup> 2000, you may want to perform several configuration tasks, for example:

- Set server configuration parameters.
- Set security parameters.
- Register the server and add it to a server group.

Use SQL Server Enterprise Manager or Transact-SQL to perform these tasks.

#### See Also

Administering SQL Server Overview

# Switching Between SQL Server 6.5 and SQL Server 2000

Microsoft® SQL Server<sup>™</sup> 2000 can be installed on the same computer with Microsoft SQL Server version 6.5, but only one version can be active at one time.

**Note** Switching between versions is not the same as running multiple instances. For more information, see <u>Working with Instances and Versions of SQL Server</u>.

After the SQL Server Upgrade Wizard finishes the conversion from SQL Server version 6.5, SQL Server 2000 is the active version of SQL Server. If enough disk space exists on your computer, you can keep the SQL Server 6.5 installation intact.

To switch from one version to the other, use the **Microsoft SQL Server-Switch** application on the **Start** menu, or run Vswitch.exe.

**IMPORTANT** Be sure the SQL Server Upgrade Wizard is finished with its upgrade tasks, before you attempt to switch between the active and nonactive versions of SQL Server.

To switch from SQL Server 6.5 to SQL Server 2000

# **Removing SQL Server 7.0 or SQL Server 6.5 After Upgrading**

After you upgrade your Microsoft® SQL Server<sup>™</sup> version 7.0 installation to Microsoft SQL Server 2000, or after you upgrade your databases from SQL Server version 6.5 to SQL Server 2000, you can keep the earlier versions on your computer.

A number of configurations are available for keeping earlier versions of SQL Server alongside SQL Server 2000. For more information, see <u>Working with</u> <u>Instances and Versions of SQL Server</u>.

When you are ready to remove an earlier version, you can use the Add/Remove Programs option in Control Panel, or you can run uninstall from the **Start** menu.

#### To remove SQL Server 7.0

• On the **Start** menu, in the SQL Server 7.0 group, click **Uninstall SQL Server 7.0**.

#### To remove SQL Server 6.5

• On the **Start** menu, in the SQL Server 6.5 group, click **Remove SQL Server 6.5**.

# **Removing SQL Server 2000**

You can remove instances of Microsoft® SQL Server<sup>™</sup> 2000 by:

- Running SQL Server 2000 Setup and selecting the Uninstall option.
- Running the Add/Remove Programs application in Control Panel.

Each named instance of SQL Server 2000 must be removed separately. You cannot remove individual components of SQL Server 2000. To remove components, you must remove the entire instance.

**IMPORTANT** Before removing SQL Server 2000, quit all applications, including the Windows NT Event Viewer, the Registry editor, all SQL Server applications, and all applications dependent on SQL Server.

#### To remove SQL Server using Control Panel

# **Upgrading to SQL Server 2000: Overview**

Upgrading from Microsoft® SQL Server<sup>TM</sup> version 7.0 to Microsoft SQL Server 2000 is one of the basic choices offered by the SQL Server Setup program on the initial **Installation Selection** screen. When you select the option to **Upgrade**, **remove**, **or add components to an existing installation of SQL Server**, Setup detects your current installation and initiates the correct sequence of setup screens for the upgrade selected. Upgrade variations include:

- A complete installation upgrade from SQL Server 7.0 to SQL Server 2000 (installing over SQL Server 7.0).
- Adding components to an installation of SQL Server 2000.
- An upgrade to the feature set of an existing installation of SQL Server 2000 (edition and component upgrade).
- An upgrade to SQL Server 2000 from SQL Server version 6.5 using the SQL Server Upgrade Wizard.
- An online database upgrade of SQL Server 7.0 databases to SQL Server 2000 database format using the Copy Database Wizard.

During the upgrade from SQL Server 7.0, external packages, such as Microsoft Management Console and the Microsoft Distributed Transaction Coordinator, must be installed for each upgrade, and the registry updated. The **master** database and other system databases are upgraded in various ways involving a series of scripts run on the server with specific options. If the upgrade process fails built-in recovery mechanisms restart and resume the upgrade.

#### See Also

Upgrading from SQL Server 7.0 to SQL Server 2000

Upgrading an Existing Installation of SQL Server Upgrading Databases from SQL Server 7.0 (Copy Database Wizard) Upgrading to a SQL Server 2000 Failover Cluster

# Hardware and Software Requirements for Upgrading

In addition to the hardware and software requirements for an installation of Microsoft® SQL Server<sup>TM</sup>, the computer must meet these requirements for an upgrade.

Hardware/software	Upgrade requirements
Operating system	Microsoft Windows NT® Server Enterprise Edition version 4.0 with Service Pack 5 (SP5) or later.
	Windows NT Server version 4.0 with SP5 or later.
	Windows NT Workstation 4.0 with SP5 or later.
	Internet Explorer 5.0 or later.
	Windows 2000.
SQL Server 6.5	When upgrading SQL Server version 6.5 to an instance of SQL Server 2000 on the same computer, you must have applied SQL Server 6.5 Service Pack 5 (SP5) or later. When upgrading SQL Server 6.5 to an instance of SQL Server 2000 on a different computer, you must have applied SQL Server 6.5 Service Pack 3 (SP3) or later.
SQL Server 7.0	SQL Server 7.0 (at any Service Pack level).
Network protocols	Named Pipes.
	SQL Server 6.5, SQL Server 7.0, and SQL Server 2000 all must be set to listen to the default pipe, \\.\pipe\sql\query. Named Pipes is required even for a tape backup upgrade.
Hard-disk space	No additional hard-disk space is required when upgrading from SQL Server 7.0 to SQL Server 2000.
	When upgrading from SQL Server 6.5 to SQL Server 2000, however, you need approximately 1.5 times the size of the SQL Server 6.5 databases.

## See Also

Hardware and Software Requirements for Installing SQL Server 2000

# **Upgrading from SQL Server 7.0 to SQL Server 2000**

You can overwrite an installation of Microsoft® SQL Server<sup>™</sup> version 7.0 with a version upgrade to Microsoft SQL Server 2000. If SQL Server 7.0 is detected as an existing installation when you run Setup, you can choose the option to upgrade. In this process, all the SQL Server 7.0 program files are upgraded, and all data stored in SQL Server 7.0 databases is preserved. In addition, SQL Server Books Online for SQL Server 7.0 remains on your computer.

**Note** SQL Server 7.0 profiler traces and registered servers are not upgraded when SQL Server 7.0 tools are upgraded to SQL Server 2000. Similarly, information models that were installed with Microsoft Repository 2.0 are not upgraded automatically. SQL Server 2000 supports newer versions of information models for both Data Transformation Services (DTS) and the Open Information Model (OIM). For more information about upgrading the DTS information model, see <u>DTS Information Model</u>. For more information about upgrading the OIM, see <u>Upgrading an Information Model</u>.

You can also upgrade from one edition of SQL Server to another edition during the version upgrade to SQL Server 2000. For more information, see <u>SQL Server</u> 2000: Editions and Components.

**CAUTION** After you perform this version upgrade, the SQL Server 7.0 installation no longer exists on your computer. The only way to restore an installation of SQL Server 7.0 is to first uninstall SQL Server 2000, perform a complete reinstall of SQL Server 7.0 files, and then restore your backed-up SQL Server 7.0 databases.

#### To upgrade an installation of SQL Server 7.0 to SQL Server 2000

# **Replication and Upgrading**

When upgrading to Microsoft® SQL Server<sup>™</sup> 2000, you can upgrade servers in your organization one at a time; however, when servers are used for replication, you must upgrade the Distributor first, the Publisher second, and then Subscribers. Upgrading servers one at a time following this sequence is recommended when a large number of Publishers and Subscribers exist because you can continue to replicate data even though servers are running different versions of SQL Server. You can create new publications and subscriptions with servers running instances of SQL Server 2000, and still maintain subscriptions created in SQL Server 6.5 or SQL Server 7.0.

When using transactional replication, you can upgrade Subscribers before the Publisher. If you are using immediate updating with snapshot replication or transactional replication, there are additional upgrade recommendations in this topic under Upgrading and Immediate Updating.

You can upgrade replication servers running SQL Server 6.5 or SQL Server 7.0 to SQL Server 2000. If the server is running SQL Server 6.5, you do not need to upgrade it to SQL Server 7.0 before upgrading to SQL Server 2000.

**IMPORTANT** When upgrading servers configured for replication to SQL Server 2000, the database compatibility level must be set to 70 (version 7.0 compatibility) or later. If you have servers running in 65 (version 6.5) or an earlier compatibility level, temporarily change them to 70 or later during the upgrade process.

When the Publisher or Subscriber is running in 65 or an earlier compatibility level during upgrade to SQL Server 2000, error 15048 will be raised stating that the operation is supported only on SQL Server version 7.0 or SQL Server 2000.

For more information about setting the backward compatibility level, see <u>SQL</u> <u>Server 2000 and SQL Server version 6.5</u>.

If you are upgrading replication on a failover cluster, you must uncluster the previous installation before upgrading. Unclustering the previous installation means that you must delete all publications, remove replication, and reconfigure it after upgrading to SQL Server 2000. This will not be a requirement when

upgrading SQL Server 2000 to future releases.

## **Upgrading and Immediate Updating**

If you are using immediate updating with snapshot replication or transactional replication, changes to that feature in SQL Server 2000 will affect how you upgrade. Rows in immediate updating articles now use a **uniqueidentifier** column to identify versions, whereas in SQL Server 7.0, a **timestamp** column was used. In addition, the triggers generated for immediate updating have been changed, and the trigger generation code has been modified to accommodate queued updating. Because of these changes, additional upgrade steps are necessary.

If using immediate updating:

- Upgrade both the Publisher and Subscriber before replicating data.
- Drop the publication and all subscriptions to the publication.
- Use an ALTER TABLE DROP COLUMN Transact-SQL statement to drop the **timestamp** column from the tables on the Publisher and from the tables on the Subscriber that allow Subscriber updates.
- Re-create the publication and subscriptions. The system adds a **uniqueidentifier** column to the published table. That column is used for row versioning (to detect conflicts when receiving updates from the Subscriber).

Although it is recommended you upgrade both the Publisher and the Subscriber and then drop and re-create the existing publications, the Publisher and Subscribers can be upgraded in any order. If you need to reinitialize a Subscriber or add a new Subscriber, you need to drop and re-create the publication.

## **Upgrading and File Transfer Protocol**

If using File Transfer Protocol (FTP), you should follow the recommended upgrade path, which ensures that Subscribers are able to obtain the necessary

FTP information from the Distributor.

SQL Server 2000 stores FTP parameters as Publication Properties; you no longer need to administer them at the Subscriber for each subscription. When upgrading to SQL Server 2000, the FTP option in the Publication Properties is turned off, and you need to open the properties for each publication that uses FTP, and then reset the FTP parameters.

SQL Server 7.0 Subscribers will continue to locate FTP files using the FTP parameters stored in the Subscription Properties when using a Distributor running an instance of SQL Server 2000. However, Subscribers running an instance of SQL Server 2000 will not be able to obtain FTP information from Distributors running earlier versions of SQL Server.

Existing subscriptions using merge replication or transactional replication will be unaffected by this change unless you need to reinitialize or connect to the FTP site. The FTP parameters need to be specified before snapshot replication occurs, or replication agents will not be able to locate the snapshot files.

For more information about changing the FTP parameters, see <u>Using TCP/IP and</u> <u>FTP</u> and <u>How to specify FTP information (Enterprise Manager)</u>.

## **Troubleshooting and Replication Upgrades**

If errors occur while upgrading replication servers, they might be related to the database being offline or unavailable or a script may have failed. For more information about troubleshooting errors that occur when upgrading replication, see <u>Help with Replication</u>.

It is recommended that you stop all data modifications at the replication server while it is being upgraded. When upgrading from SQL Server 6.5, you must run the Log Reader Agent and Distribution Agent before upgrading to make sure there are no replicated commands pending delivery to Subscribers.

Because you can upgrade servers running instances of Microsoft® SQL Server<sup>TM</sup> 2000 one at a time, you may have circumstances where servers in your replication topology are running different versions of SQL Server. You can replicate between different versions of SQL Server, but you are often limited to the functionality of the earliest version used.

**IMPORTANT** When upgrading from SQL Server 6.5 or 7.0 to SQL Server 2000,

SQL Server Setup runs several \*.sql replication scripts. Although the upgrade process can take several minutes and does not display progress notifications, you can view error messages in the \*.out and \*.err files located in the SQL Server Install directory.

## See Also

Publishing Data Over the Internet Using TCP/IP and FTP Replication Between Different Versions of SQL Server Replication Data Considerations Updatable Subscriptions

# **Upgrading Databases from SQL Server 7.0 (Copy Database Wizard)**

As an enhancement to the regular upgrade procedure, you can perform an online upgrade of databases and associated meta data. Using the Copy Database Wizard, you can move or copy a database from Microsoft® SQL Server<sup>™</sup> 7.0 to an instance of Microsoft SQL Server 2000, without having to shut down any servers in the process.

Advantages of an online database upgrade include:

- No downtime for servers during the upgrade.
- Custom selection of databases to upgrade, leaving other databases still available to the original (SQL Server 7.0) server.
- Inclusion of related meta data in the upgrade procedure. For example, logon information, jobs, and user-specific objects associated with user databases can be included.
- The process can be run at a convenient time.

The Database Copy Wizard is based on detach and attach functionality that allows user databases to be moved or copied from a source to a destination server. A Data Transformation Services (DTS) package performs the actual move or copy operation You can schedule the package to run at a specified time or rerun the package if required.

## **Options for SQL Server 7.0 Database Upgrades**

Database administrators can move or copy one or more databases from an instance of SQL Server 7.0 to the default instance of SQL Server 2000 on your local computer, or to either a default or a named instance on a remote computer. This upgrade feature does not support SQL Server 6.5 databases.

#### Local computer

• SQL Server 7.0 databases can be upgraded to a named instance of SQL Server 2000 on the local computer.

#### **Remote computer**

- SQL Server 7.0 databases can be upgraded to a default instance of SQL Server 2000 on a remote computer.
- SQL Server 7.0 databases can be upgraded to a named instance of SQL Server 2000 on a remote computer.

**Note** You can have only one active default instance of SQL Server on a computer at one time; either a default instance of SQL Server 7.0 or a default instance of SQL Server 2000. SQL Server 6.5 can also be a default instance. For more information, see <u>Working with Instances and Versions of SQL Server</u>.

#### Exceptions

The Copy Database Wizard cannot be used in these situations:

- A database with the identical name on both source and destination servers cannot be moved or copied. On the database selection screen, it will be noted as "Already exists."
- For databases involved in replication, a regular server upgrade is required.

## **Copy Database Wizard Safeguards**

At the start of a database move or copy operation, one administrator must have exclusive use of all files to prevent any changes to the file set during the process. Two connections are required to copy database files: **sysadmin** privileges on both installations of SQL Server and administrator privileges on the server/network.

To prevent any chance of data corruption, the SQL Server 7.0 databases must be

in read-only condition and cannot be renamed during this operation. Any name conflicts between source and destination servers must be resolved manually prior to upgrading databases. Nothing on the destination server is overwritten.

If you move or copy multiple databases in one operation, each database is actually moved one at a time; that is, one database at a time is detached, files are copied and then reattached. To avoid any problems, the DTS package writes a message to the error log indicating that the database is about to be detached from its source server. At the same time, a script is prepared to attach the database to its destination. After the database is successfully attached to the destination, another entry is written to the log indicating successful completion.

When upgrading to a destination that is a clustered server, the Copy Database Wizard will ensure you select only shared drives on a clustered destination server. The source server may also be clustered.

**Note** Unrelated to this upgrade process, you can also use the Copy Database Wizard to move or copy user databases from one instance of SQL Server 2000 to another instance of SQL Server 2000. For more information, see <u>Using the Copy</u> <u>Database Wizard</u>.

#### To upgrade databases online using the Copy Database Wizard

# **Upgrading Databases from SQL Server 6.5 (Upgrade Wizard)**

You can convert data from Microsoft® SQL Server<sup>™</sup> version 6.5 to the formats for SQL Server 2000 using the SQL Server Upgrade Wizard. The wizard upgrades any or all of your databases, transferring all catalog data, objects, and user data. It also transfers replication settings, SQL Executive settings, and most of the SQL Server 6.5 configuration options. Be sure to review all aspects of this upgrade, as noted in <u>Preparing to Upgrade from SQL Server 6.5</u>.

**Note** To run the SQL Server Upgrade Wizard, you must have a default instance of Microsoft SQL Server 2000 installed on your computer.

The SQL Server Upgrade Wizard does not support consolidation of databases from multiple SQL Server 6.5 installations. If you must upgrade SQL Server 6.5 databases from multiple servers, consolidate all of the SQL Server 6.5 databases onto one server, and then run the wizard to upgrade the consolidated server.

The SQL Server Upgrade Wizard does not remove SQL Server 6.5 from your computer. If you are using a tape backup to perform the upgrade, you have the option of removing the SQL Server 6.5 devices to save disk space.

When the upgrade process is complete, two separate installations of SQL Server exist, including two separate sets of the same data. The SQL Server 6.5 and the SQL Server 2000 installations become independent of each other.

If you are performing the upgrade on a single computer, additional disk space is required. For more information, see <u>Estimating the Disk Space Required for</u> <u>Upgrading</u>. You can also upgrade from one computer to another. For more information, see <u>Upgrading Using One or Two Computers (Logon Screen)</u>.

**Note** You can leave the installation of SQL Server 6.5 on a computer indefinitely. In addition to installations of SQL Server 6.5 and a default instance of SQL Server 2000 on the same computer, you also can install multiple named instances of the SQL Server 2000 on the same computer. For more information, see <u>Working with Instances and Versions of SQL Server</u>.

#### To perform a version upgrade using a direct pipeline

## **Preparing to Upgrade from SQL Server 6.5**

Follow this checklist before using the SQL Server Upgrade Wizard to move from Microsoft® SQL Server<sup>™</sup> version 6.5 to Microsoft SQL Server 2000:

- Back up the SQL Server 6.5 database files (all .dat files, including **master**) so you can completely restore them if necessary.
- Run the appropriate Database Console Commands (DBCC) on the SQL Server 6.5 databases to ensure they are in a consistent state.
- Estimate the disk space required. In addition to the hard disk space used by Microsoft SQL Server 2000, you need approximately 1.5 times the size of the SQL Server 6.5 databases.
- Set **tempdb** to at least 10 MB in the SQL Server 6.5 installation, 25 MB is recommended.
- Ensure the **master** database has at least 3 MB of free space.
- Ensure that all database users have logon information in the **master** database.

This is important for restoring a database because system logon information resides in the **master** database.

• Ensure the @@SERVERNAME is defined on SQL Server 2000. If @@SERVERNAME is NULL, you can use the sp\_addserver system stored procedure. For example, if your computer is named production, the command would be sp\_addserver 'production1',local. Changes do not take affect until the MSSQLServer service is restarted.

**Note** Because SQL Server 6.5 does not recognize the hyphen (-) in a computer name, replace a hyphen with an underscore (\_).

• Disable any startup stored procedures.

The SQL Server Upgrade Wizard starts and stops the SQL Server 6.5 server during the upgrade process. Stored procedures processed at startup may cause the upgrade process to stop responding.

• Ensure that you upgrade all databases with cross-database dependencies at the same time.

For example, you want to upgrade three databases, **database1**, **database2**, and **database4**, and there is logon information in SQL Server 6.5 **master..sysdatabases** for **USER1** that defaults to **database3** (not one of the databases you are upgrading). The SQL Server Upgrade Wizard does not create the logon information because the database is not upgraded, and therefore does not exist in SQL Server 2000. If **USER1** is listed as the owner for objects in any of the databases upgraded, those objects cannot be created because the logon information for **USER1** does not exist.

- If performing a two-computer upgrade, assign a domain user name and password to the MSSQLServer service for SQL Server 6.5 and SQL Server 2000 instead of using the local system account or a local user account. The domain user account should belong to the **Administrators** group of both the computers involved in the upgrade. (The local system account is sufficient for a one-computer upgrade.)
- Stop replication and ensure that the log is empty.
- Quit all applications, including all services dependent on SQL Server.

If you copied the SQL Server 6.5 databases to a new computer to perform the upgrade, you may need to update the new SQL Server 6.5 **master** database as follows:

- Change references from the earlier server name to the current server name in the SQL Server 6.5 **master** database.
- Update the device file locations in the SQL Server 6.5 **master** database.

• Ensure all users have corresponding logon information.

To change the size of tempdb in SQL Server 6.5

# **Estimating the Disk Space Required for Upgrading**

Before you perform an upgrade of Microsoft® SQL Server<sup>™</sup> version 6.5 to SQL Server 2000, ensure that there is available disk space. This is important if you intend to perform either a one-computer or a two-computer upgrade.

The SQL Server Upgrade Wizard estimates the disk space necessary to upgrade the SQL Server 6.5 server to SQL Server 2000. The wizard examines the current SQL Server 6.5 installation and estimates the amount of disk space the SQL Server 6.5 data will occupy in SQL Server 2000.

You can estimate:

- The size of SQL Server 2000 databases.
- The size of SQL Server 2000 logs.
- The amount of disk space required for **tempdb**.

**Note** The SQL Server Upgrade Wizard estimates the disk space required; it cannot give an exact requirement.

To estimate the disk space required for an upgrade

## **Data and Object Transfer**

The **Data and Object Transfer** screen allows you to choose upgrade options.

## Export from 6.5 Server / Import

The objects and data check boxes indicate that the SQL Server Upgrade Wizard exports catalog data, objects, and user data from selected Microsoft® SQL Server<sup>™</sup> version 6.5 databases and imports them into newly created SQL Server 2000 databases.

## Data Transfer Method

You can perform an upgrade using either of the following data transfer methods:

• Named pipe (simultaneous import/export)

A direct pipeline enables the SQL Server Upgrade Wizard to transfer data in memory from Microsoft SQL Server version 6.5. This data transfer method is the most reliable and provides the best performance. However, when performing a one-computer upgrade, you cannot reuse the disk space occupied by the SQL Server 6.5 devices until the version upgrade process is complete, so use this option only if you have disk space available.

• **Tape** (requires a Microsoft Windows NT<sup>®</sup> tape driver to be installed)

The SQL Server Upgrade Wizard backs up to tape all of the SQL Server 6.5 databases you have selected to upgrade. The SQL Server Upgrade Wizard then optionally deletes all of the SQL Server 6.5 devices, freeing disk space before new data files are created.

**IMPORTANT** The SQL Server Upgrade Wizard deletes all of the SQL Server 6.5 devices, not only those upgraded. You should upgrade all databases if you choose to delete the SQL Server 6.5 devices.

The tape backup option should be used only when you want to upgrade on a single computer but there is not enough space on the hard disk to install SQL Server 2000 alongside SQL Server 6.5 and perform the version upgrade.

**Note** The SQL Server Upgrade Wizard uses a named pipe, even when performing a tape backup upgrade. SQL Server 6.5 and SQL Server 2000 must be set to listen to the default named pipe, \\.\pipe\sql\query.

#### Verification

The transfer of objects and data by the SQL Server Upgrade Wizard is a very reliable process. If any objects could not be imported due to errors in those objects or compatibility problems with Microsoft SQL Server, they are noted in the output logs of the SQL Server Upgrade Wizard.

The SQL Server Upgrade Wizard also offers the following optional verification measures:

#### • Validate successful object data transfer

The SQL Server Upgrade Wizard examines the SQL Server 6.5 databases before the upgrade process and SQL Server 2000 databases after the upgrade. For each, the wizard prepares a list of all objects, including schema and stored procedures, and the number of rows in each table. The wizard then compares the two lists and reports any discrepancies.

#### • Exhaustive data integrity verification

The SQL Server Upgrade Wizard performs a checksum for each column of each table before and after the upgrade to verify that data values have not changed.

**Note** The SQL Server Upgrade Wizard does not report as errors any intentional differences in objects. If some objects, typically stored procedures, could not import due to errors in the objects or compatibility problems with SQL Server 2000, they are reported twice: once in the SQL scripts that show the source code of the objects and the error messages received from SQL Server 2000 when trying to create them, and then again in the output of the verification processes.

# Order of Upgrade Using a Direct Pipeline or Tape Drive

The SQL Server Upgrade Wizard performs a version upgrade using the options specified. The Microsoft® SQL Server<sup>™</sup> version 6.5 server and data used by SQL Server 6.5 databases are left intact throughout the version upgrade process. At this time, the SQL Server 6.5 catalog data, objects, and databases are converted so that they are compatible with SQL Server 2000. After the version upgrade is complete, SQL Server 2000 becomes your production system.

The order of upgrade is basically the same for both a direct pipeline and a tape drive upgrade. The one difference is in how data is exported and imported. When using a tape drive, data is exported to the tape drive after shutting down SQL Server 6.5 and before starting SQL Server 2000. This data is then imported from the tape drive later to SQL Server 2000. When using a direct pipeline, the export and import steps are combined in one step, simultaneously.

The following list shows the order in which the SQL Server Upgrade Wizard performs the upgrade from SQL Server 6.5 to SQL Server 2000. The differences between the direct pipeline and tape drive methods are noted.

- Starts SQL Server 6.5
- Updates ODBC and SQL-DMO components on SQL Server 6.5
- Examines SQL Server 6.5 databases
- Exports replication settings
- Exports server configuration settings from the **master** database
- Exports logon information

- Exports database owners
- Exports SQL Executive objects and settings from the **msdb** database
- Exports database objects for all databases chosen
- Shuts down SQL Server 6.5
  - Tape Drive only: Exports data to tape
  - Tape Drive only: Backs up and then deletes SQL Server 6.5 devices
- Starts SQL Server 2000
- Creates databases
- Modifies SQL Executive objects and settings to SQL Server 2000 formats
- Imports logon information
- Imports database objects
  - Tape Drive only: Imports data from tape into SQL Server 2000
  - Direct Pipeline only: Simultaneously exports data from SQL Server 6.5 and imports it into SQL Server 2000
- Imports modified SQL Executive objects and settings into SQL Server 2000
- Imports replication settings

- Examines SQL Server 2000 databases
- Verifies that the upgrade is successful
- Sets database options in SQL Server 2000
- Marks server and databases as moved
- Drops temporary **tempdb** files

# Upgrading Using One or Two Computers (Logon Screen)

The upgrade process can take place on a single computer or from one computer to another, depending on where Microsoft® SQL Server<sup>™</sup> version 6.5 and SQL Server 2000 are installed. The SQL Server Upgrade Wizard identifies the two servers as the export server and import server.

- For a one-computer upgrade, leave the import and export servers at their default values.
- For a two-computer upgrade, select the name of the computer with your SQL Server 6.5 server as the export server. To upgrade SQL Server from one computer to another, the two computers must be in the same network domain.

**IMPORTANT** The one-computer upgrade is the only method supported when upgrading a server used in replication. A two-computer upgrade is not supported for replication servers.

## Export server (6.5)

Export server (6.5) is the name of the SQL Server 6.5 server. This defaults to the name of the computer on which the SQL Server Upgrade Wizard is run, but may be changed if your SQL Server 6.5 server is on another computer.

#### • Server name

Server name is the name of your SQL Server version 6.5 server. This defaults to the name of the computer on which the SQL Server Upgrade Wizard is run, but may be changed if your SQL Server 6.5 server is on another computer.

#### • Administrator password ('sa')

Enter the system administrator (**sa**) password for the SQL Server 6.5 server.

#### • Optional startup arguments

Enter any trace flags or other startup parameters to be used when the SQL Server Upgrade Wizard starts the SQL Server 6.5 server.

#### **Import server**

The import server is the name of the SQL Server 2000 server. This is always the name of the computer on which the SQL Server Upgrade Wizard is run.

#### • Server name

Server name is the name of your SQL Server 2000 server computer. This is always the name of the computer on which the SQL Server Upgrade Wizard is run.

#### • Administrator password ('sa')

Enter the system administrator (**sa**) password for the SQL Server 2000 server. Unless you have changed it since installing SQL Server 2000, the default **sa** password is blank.

#### • Optional startup arguments

Enter any trace flags or other startup parameters to be used when the SQL Server Upgrade Wizard starts the SQL Server 2000 server.

# **Selecting a Scripting Code Page**

The SQL Server Upgrade Wizard requires the selection of a scripting code page, which is used to create the upgrade scripts. When the **Code Page Selection** screen appears in the Upgrade Wizard, most users can accept the default code page, which is the code page recorded in the **master** database.

In some cases, the actual code page used for a Microsoft® SQL Server<sup>TM</sup> 6.5 installation differs from the code page recorded in the **master** database. If you know that the actual code page is different from the recorded code page, select the actual code page in the list on the **Code Page Selection** screen.

**CAUTION** If you choose a scripting code page other than the default, do not upgrade replication settings. If the server is involved in replication, reconfigure the replication settings after the upgrade is complete.

The enhancements to collation settings in SQL Server 2000 do not apply directly to this selection of a code page for the SQL Server 6.5 upgrade. For more information about collation enhancements, see <u>Collations</u>.

# **Selecting Databases to Upgrade**

When running the SQL Server Upgrade Wizard, you can choose to upgrade some or all Microsoft® SQL Server<sup>™</sup> version 6.5 databases. The **master**, **msdb**, and **publication** system databases, as well as the **pubs** and **Northwind** sample databases, are not explicitly available for selection. However, the **master**, **msdb**, and **publication** databases can be selected for upgrading (the default) in the **Server Configuration** dialog box of the SQL Server Upgrade Wizard.

**Note** If you run the SQL Server Upgrade Wizard again after databases have been upgraded, previously updated databases will default to the excluded list. If you want to upgrade a database again, drop the database in SQL Server 2000 and move it to the included list in the wizard.

# **Database Configuration**

Before any data is transferred, the SQL Server Upgrade Wizard creates, if necessary, database and log files large enough to contain the upgraded database data. On the **Database Creation** screen there are several options for creating the Microsoft® SQL Server<sup>TM</sup> 2000 database and log files.

## **Using the Default Database Configuration**

The SQL Server Upgrade Wizard estimates how much disk space is necessary to hold transferred objects and data for each selected database and creates database files of the estimated sizes. The wizard makes no allowance for free space beyond the loaded data. By default, the data file for a database is placed in the same location as the first device used by that database in SQL Server 6.5.

The SQL Server Upgrade Wizard also creates a log file for each database using the SQL Server 6.5 log size. By default, the log file is placed in the same location as the first device used for log space in SQL Server 6.5.

You can view and edit the default database configuration in the SQL Server Upgrade Wizard. For each database and log file you can modify:

- The name and file path.
- The initial size of the file.
- The autogrow increment.

If using multiple devices in a SQL Server version 6.5 database, then multiple database files are created in the same location. However, the first database file is sized to accommodate the bulk of the data, and the other files are minimally sized. If you want to remove these files, you must do so before they are created. All files are set to grow automatically if extra space is required.

## **Using a Custom Database Configuration**

You can specify a custom configuration in two ways:

• Using databases and logs that you created in SQL Server 2000.

The SQL Server Upgrade Wizard does not create any user databases. You must create the necessary databases and logs in SQL Server 2000 before you start the SQL Server Upgrade Wizard. Use this option only if necessary.

• Using an SQL script file that you provide.

The SQL Server Upgrade Wizard uses an SQL script file that you provide to create the necessary user databases and logs. Use this option only if you are familiar with the new CREATE DATABASE statement in SQL Server 2000.

If you create the user databases or an SQL script file, the SQL Server 2000 databases must have the same names as in SQL Server 6.5. Also, remember that data may take up more disk space in SQL Server 2000 than in SQL Server 6.5. The SQL Server Upgrade Wizard estimates this growth. You can view the proposed layout of the SQL Server 2000 data files to see the estimated initial size of the SQL Server 2000 database, and edit the default configuration, if necessary. For more information, see <u>Proposed Database Layout</u>.

It is recommended that you leave the autogrow feature on for each database. You may also want to set a backward compatibility level for each database.

#### To edit the default database configuration

# **Proposed Database Layout**

The **Proposed Database Layout** dialog box lists the databases, file groups, and data files that the Microsoft® SQL Server<sup>™</sup> Upgrade Wizard will create. You can create or remove file groups and data files from the **File** menu. Double-click a data file to edit the file name, initial size, or file growth details.

#### **Object Details**

Click on a file group or data file in the proposed database layout to view details. Click a database in the proposed database layout to view summary information.

#### **Drive Summary**

The drive summary lists all local fixed-disk drives. For each drive, the existing SQL Server version 6.5 data file size, proposed SQL Server 2000 data file size, and free space are listed. On the **Options** menu, select **Freespace includes 6.5 files** to view the free space that would exist if the SQL Server 6.5 data files were deleted. This option shows the disk space available if the upgrade is performed using tape and the SQL Server 6.5 devices are deleted.

#### See Also

**Database** Configuration

# **Tape Upgrade Transfer Options**

When you perform a tape backup, you must select a tape drive and choose how the SQL Server Upgrade Wizard handles backing up and deleting objects in the Microsoft® SQL Server<sup>TM</sup> version 6.5 databases.

#### **Device for Data Transfer**

The SQL Server Upgrade Wizard transfers all of the data you are upgrading to this tape drive before the SQL Server 2000 databases are created.

### **Backing Up the SQL Server 6.5 Devices**

You may also choose to back up the SQL Server 6.5 devices. This is separate from the transfer to tape that the SQL Server Upgrade Wizard uses to complete the upgrade. There are two options for backing up the devices:

#### • Prompt me to backup my devices manually

Before data is exported, the SQL Server Upgrade Wizard pauses and prompts you to perform a backup. The SQL Server Upgrade Wizard does not perform a backup for you. You must use a backup utility such as Microsoft Windows NT® Backup.

#### • Automatically copy device files to the following location

Before data is exported, the SQL Server Upgrade Wizard copies the device files to a shared network directory.

**WARNING** If you back up the devices to tape, remove the tape backup and insert a blank tape before continuing. Before the SQL Server Upgrade Wizard begins transferring data to the tape drive, it formats the tape in the drive. If you do not remove your tape backup, the SQL Server Upgrade Wizard overwrites it.

#### **Deleting the SQL Server 6.5 Devices**

If you decide to delete your SQL Server 6.5 devices before creating the SQL Server 2000 databases, you can choose whether to be prompted before the

devices are deleted. All of the SQL Server 6.5 device files will be deleted if you choose to delete devices, even if you are upgrading only one database. This will render the SQL Server 6.5 server unusable until the files are restored.

**Note** If you choose not to delete the devices, you must have enough disk space for both the SQL Server 6.5 and SQL Server 2000 databases. If sufficient space is available, you should use a Named Pipe upgrade instead of a Tape upgrade.

#### See Also

<u>How to perform a SQL Server version 6.5 to SQL Server 2000 upgrade using a tape drive (SQL Server Upgrade Wizard)</u>

# **System Configuration**

On the **System Configuration** screen, you can set options for system objects to transfer, ANSI Nulls, and quoted identifiers.

#### System Objects to Transfer

When the SQL Server Upgrade Wizard upgrades the **master** database, it can upgrade several configuration options:

• Server configuration

Logon information and remote logon registrations and server configuration options relevant to Microsoft® SQL Server<sup>™</sup> 2000 are transferred as part of the version upgrade process. The SQL Server 6.5 configuration options not used in SQL Server 2000 are not transferred.

• Replication settings

All articles, subscriptions, and publications of each selected database, including the distribution database, if any, are transferred and upgraded.

• SQL Executive settings

All tasks scheduled by SQL Executive are transferred and upgraded so that the SQL Server 2000 can schedule and run the tasks in SQL Server Agent.

**Note** Upgrading replication or SQL Executive settings causes existing modifications made to the SQL Server 2000 replication or SQL Server Agent settings to be overwritten.

#### **ANSI Nulls**

The ANSI\_NULLS option controls both database default nullability and comparisons against null values. When upgrading Microsoft SQL Server version 6.5 to the SQL Server 2000, set the ANSI\_NULLS option to ON or OFF.

When the SQL Server Upgrade Wizard creates the SQL Server 2000 database tables, the database default nullability determined by the ANSI\_NULLS option

is not an issue. All columns are explicitly qualified as NULL or NOT NULL based on their status in SQL Server 6.5.

The ANSI\_NULLS option is important with regard to comparisons against null values, when the SQL Server Upgrade Wizard creates the SQL Server 2000 database objects. With ANSI\_NULLS set to ON, the comparison operators EQUAL (=) and NOT EQUAL (<>) always return NULL when one of its arguments is NULL. With ANSI\_NULLS set to OFF, these operators return TRUE or FALSE, depending on whether both arguments are NULL.

In SQL Server 6.5, the ANSI\_NULLS option in objects, such as stored procedures and triggers, is resolved during query execution time. In SQL Server 2000, the ANSI\_NULLS option is resolved when the object is created. You must choose the ANSI\_NULLS option setting you want for all objects in the databases you are upgrading. The SQL Server Upgrade Wizard then creates all database objects using this ANSI\_NULLS setting.

# **Quoted Identifiers**

**Note** Quoted identifiers are used by default in SQL Server 2000, that is, they are set to ON. This is different from SQL Server 7.0 where they were set to OFF by default.

The QUOTED\_IDENTIFIER setting determines what meaning Microsoft SQL Server gives to double quotation marks ("). When QUOTED\_IDENTIFIER is set to OFF, double quotation marks delimit a character string, just as single quotation marks do. When QUOTED\_IDENTIFIER is set to ON, double quotation marks delimit an identifier, such as a column name. An identifier must be enclosed in double quotation marks; for example, if its name contains characters that are otherwise not allowed in an identifier, including spaces and punctuation, or if the name conflicts with a reserved word in Transact-SQL. Regardless of the QUOTED\_IDENTIFIER setting, an identifier can also be delimited by square brackets.

The meaning of the following statement, for example, depends on whether QUOTED\_IDENTIFIER is set to ON or OFF:

#### SELECT "x" FROM T

If QUOTED\_IDENTIFIER is set to ON, "x" is interpreted to mean the column

named **x**. If it is set to OFF, "x" is the constant string x and is equivalent to the letter x.

If the previous SELECT statement example were part of a stored procedure created when QUOTED\_IDENTIFIER was set to ON, then "x" would always mean the column named **x**. Even if the QUOTED\_IDENTIFIER setting was later switched, and set to OFF, the stored procedure would respond as if it were set to ON and treat "x" as the column named **x**.

When the SQL Server Upgrade Wizard re-creates database objects in SQL Server 2000, the QUOTED\_IDENTIFIER setting determines how all of these objects behave. If all database objects were created in SQL Server 6.5 with the same QUOTED\_IDENTIFIER setting, click that setting, either **On** or **Off**. If objects were created in SQL Server version 6.5 with a mix of the two settings, or if you are unsure of the settings used, click **Mixed**.

With the **Mixed** option, the SQL Server Upgrade Wizard first converts all objects containing double quotation marks with QUOTED\_IDENTIFIER set ON. The SQL Server Upgrade Wizard then converts any objects that failed to be created with QUOTED\_IDENTIFIER set OFF.

#### See Also

<u>How to perform a SQL Server version 6.5 to SQL Server 2000 upgrade using a</u> <u>direct pipeline (SQL Server Upgrade Wizard)</u>

<u>How to perform a SQL Server version 6.5 to SQL Server 2000 upgrade using a tape drive (SQL Server Upgrade Wizard)</u>

# **Completing the SQL Server Upgrade Wizard**

Use this screen to view the summary of choices you have made.

Click **View warnings and choices in notepad** to open a text version of the upgrade script. If all options are correct, click **Finish**.

# **Upgrade Script Interpreter**

After you click **Finish**, this screen displays the progress of the upgrade.

#### **Progress indicator**

Displays information about the current task and its progress toward completion. The information presented varies according to the type of task.

#### Task

The SQL Server Upgrade Wizard adds each upgrade task to the list as it is started.

#### Status

The SQL Server Upgrade Wizard displays the status (Running, Done, or Error) for each task.

#### Started

The SQL Server Upgrade Wizard displays the time and date on which the task began.

#### End

The SQL Server Upgrade Wizard displays the time and date on which a completed or terminated task is finished.

#### **Pause Task**

Temporarily suspends the version upgrade process until you click **Resume**.

#### **Cancel Task**

Cancels the currently running task and proceeds to the next task. Do not cancel a task unless you are certain the current task does not need to be completed before subsequent tasks are run.

#### **Retry Task**

Retries the current upgrade task. If a task ended in an error and you corrected the problem, the SQL Server Upgrade Wizard retries the current task.

#### **Pause Between Steps**

Allows you to participate interactively in the version upgrade process and track the progress of the SQL Server Upgrade Wizard. The SQL Server Upgrade Wizard asks for confirmation between each step of the version upgrade process.

# **Backward Compatibility**

Backward compatibility issues are divided in these sections:

• For issues related to upgrades from Microsoft® SQL Server<sup>™</sup> version 7.0 to SQL Server 2000, see:

SQL Server 2000 and SQL Server version 7.0

• For issues related to upgrades between SQL Server 6.5 and Microsoft SQL Server 2000, see:

SQL Server 2000 and SQL Server version 6.5

If upgrading from SQL Server 6.5 to SQL Server 2000, review both sections.

# **SQL Server 2000 and SQL Server version 7.0**

Microsoft® SQL Server<sup>™</sup> 2000 is compatible with SQL Server 7.0 in most ways. The section describes backward compatibility issues when upgrading from SQL Server 7.0 to SQL Server 2000:

- Client Network Utility and Named Instances
- Multiserver Jobs and Named Instances
- Upgrading SQL Server 6.5 Client Software
- Authentication Modes
- ROWCOUNT Setting for Operations Against Remote Tables
- Server Configuration Options
- Recovery Models and Database Options
- Reserved Keywords
- SQL Profiler Extended Stored Procedures
- Default Connection Option Settings in SQL Query Analyzer
- **bcp** Utility
- Database Diagrams from Earlier Versions of Visual Database Design Tools

- Data Transformation Services
- Specifying Trusted Connections
- Extended Objects in SQL-DMO
- SQL-SCM
- English Query and SQL Server 7.0 OLAP Services

#### **Client Network Utility and Named Instances**

When using the SQL Server client connectivity components from SQL Server 7.0 or earlier, you must set up an alias using the Client Network Utility before you connect to a named instance of SQL Server 2000. For example, on a SQL Server 7.0 client, to connect to a named instance of SQL Server 2000, you must add an alias that points to

\\computername\pipe\MSSQL\$instancename\sql\query. If you use an alias name of computername\instancename, clients can connect by specifying this name in the same way as SQL Server 2000 clients do. For the TCP/IP Sockets and NWLink IPX/SPX Net-Libraries, you must use the Client Network Utility to define an alias on the client that specifies the port address on which the named instance is listening.

#### **Multiserver Jobs and Named Instances**

When using Master Servers and Target Servers, SQL Server 7.0 cannot interoperate with named instances of SQL Server 2000. To use an instance of SQL Server 7.0 with an instance of SQL Server 2000 for MSX/TSX operations, you must use a default instance, not a named instance, of SQL Server 2000.

#### **Upgrading SQL Server 6.5 Client Software**

When running an instance of SQL Server version 6.5 on a server, this issue

applies:

If you are upgrading from SQL Server 6.5 client software to SQL Server 2000 client software (and you have an application that uses the default Net-Library), you must use the Client Network Utility to make either Named Pipes or Multiprotocol the default Net-Library to make Windows Authentication connections.

## **Authentication Modes**

SQL Server 2000 can operate in one of two security (authentication) modes:

- Windows Authentication Mode (Windows Authentication)
- Mixed Mode (Windows Authentication and SQL Server Authentication)

Mixed Mode allows users to connect using Windows Authentication or SQL Server Authentication. Users who connect through a Microsoft Windows NT® 4.0 or Windows 2000 user account can make use of trusted connections (connections validated by Windows NT 4.0 or Windows 2000) in either Windows Authentication Mode or Mixed Mode.

SQL Server Authentication is provided for backward compatibility. An example of SQL Server Authentication would be if you create a single Microsoft Windows® 2000 group, add all necessary users to that group, and then grant the Windows 2000 group login rights to SQL Server and access to any necessary databases.

## **ROWCOUNT Setting for Operations Against Remote Tables**

ROWCOUNT is not supported for INSERT statements against remote tables in SQL Server 2000 when the database compatibility level is set to 80. For these INSERT operations, the SET ROWCOUNT option is ignored.

The ROWCOUNT setting for INSERT statements against remote tables was supported in SQL Server 7.0.

## **Server Configuration Options**

These server configuration options are not supported in SQL Server 2000.

default sortorder id	resource timeout
extended memory size	spin counter
language in cache	time slice
language neutral full-text	unicode comparison style
max async IO	unicode locale id

For more information about configuration options, see <u>Setting Configuration</u> <u>Options</u> and <u>sp\_configure</u>.

### **Recovery Models and Database Options**

Microsoft® SQL Server<sup>™</sup> 2000 provides the following recovery models to simplify recovery planning, simplify backup and recovery procedures, and to clarify tradeoffs between system operational requirements:

- Simple Recovery
- Full Recovery
- Bulk-Logged Recovery

Each model addresses different needs for performance, disk and tape space, and protection against data loss.

In SQL Server 7.0 and earlier, similar functionality was provided through the combined settings of the **trunc. log on chkpt** and **select into/bulkcopy** database options, which could be set using the **sp\_dboption** stored procedure.

This table maps the settings of **trunc. log on chkpt** and **select into/bulkcopy** to the new recovery models.

If trunc. log on chkpt is:	And select into/bulkcopy is:	The recovery model is:
FALSE	FALSE	FULL

FALSE	TRUE	BULK-LOGGED
TRUE	TRUE	SIMPLE
TRUE	FALSE	SIMPLE

**Note** If you upgrade a database in which the **trunc. log on chkpt** and **select into/bulkcopy** options are set to TRUE, **select into/bulkcopy** is set to FALSE, forcing the database into the simple recovery model.

The **trunc. log on chkpt** and **select into/bulkcopy** database options are supported in SQL Server 2000 for backward compatibility purposes, but may not be supported in future releases.

In SQL Server 2000, the ALTER DATABASE Transact-SQL statement provides a SET clause for specifying database options, including recovery models. For more information about database options, see <u>Setting Database Options</u> and <u>ALTER DATABASE</u>.

### **Reserved Keywords**

These words are no longer reserved keywords in SQL Server 2000: AVG, COMMITTED, CONFIRM, CONTROLROW, COUNT, ERROREXIT, FLOPPY, ISOLATION, LEVEL, MAX, MIN, MIRROREXIT, ONCE, ONLY, PERM, PERMANENT, PIPE, PREPARE, PRIVILEGES, REPEATABLE, SERIALIZABLE, SUM, TAPE, TEMP, TEMPORARY, UNCOMMITTED, WORK.

These words are reserved keywords in SQL Server 2000: COLLATE, FUNCTION, OPENXML.

## **SQL Profiler Extended Stored Procedures**

SQL Profiler extended stored procedures, such as **xp\_trace\_addnewqueue** and **xp\_trace\_generate\_event**, are not supported in SQL Server 2000. They have been replaced by a set of new stored procedures and system user-defined functions. For more information, see <u>Creating and Managing Traces and Templates</u>.

## **Default Connection Option Settings in SQL Query Analyzer**

In SQL Server version 7.0 and earlier, the default setting for SET QUOTED\_IDENTIFIER in SQL Query Analyzer was OFF. In SQL Server 2000, the default setting in SQL Query Analyzer is ON, which is also the default setting for ODBC and OLE DB. Moreover, several new features in SQL Server 2000, such as indexed views and indexes on computed columns, require this option to be ON.

**Note** If you use double quotation marks for strings when QUOTED\_IDENTIFIER is ON, you will receive a syntax error.

## bcp Utility

To read character files created by earlier versions of DB-Library **bcp** in SQL Server 2000, use the **-V** switch. For more information, see **<u>bcp</u>** <u>Utility</u>.

# **Database Diagrams from Earlier Versions of Visual Database Design Tools**

For users who have database diagrams created with earlier versions of the visual database design tools:

• If the first visual database tool that was used against a SQL Server 2000 database is a version earlier than the tools in SQL Server 2000, SQL Server Enterprise Manager will not be able to open or create a database diagram in that database. Any attempt to do so results in the error: ODBC error: [Microsoft][ODBC SQL Server Driver][SQL Ser

There are several visual database tools that can put a database into this state. These include the Query Designer, the View Designer, the Database Designer, and the Table Designer in SQL Server 7.0 and earlier, as well as many tools that enumerate the objects in a database. These tools are also in Microsoft Access 2000 and Microsoft Visual Studio® 6.

Running the following script on the database allows SQL Server Enterprise Manager to work with the database diagrams in that database:

alter table dbo.dtproperties add uvalue nvarchar(255) null

```
go
if exists(select * from dbo.dtproperties) exec('update dbo.dtprop
go
```

After this script has been run, both the SQL Server Enterprise Manager in SQL Server 2000 and the earlier versions of the visual database tools can jointly access the database diagrams in the database. There are additional issues to consider when using the earlier versions of the database tools against a SQL Server 2000 database. For more information, see <u>Hardware and Software Requirements for Installing</u> <u>SQL Server 2000</u>.

## **Data Transformation Services**

These are the backward compatibility issues for Data Transformation Services (DTS).

## **Extended DTS Objects**

Some objects in Data Transformation Services (DTS) are extended in SQL Server 2000. For more information about using new Data Transformation Services objects, methods, and properties with SQL Server 7.0 and earlier, see <u>Extended DTS Objects</u>.

## **Copy SQL Server Objects Task**

There are restrictions on using the Copy SQL Server Objects task (Transfer SQL Server Objects task in SQL Server version 7.0) when copying database objects between an instance of SQL Server 2000 and SQL Server 7.0. For more information, see <u>Copy SQL Server Objects Task</u>.

## **Running DTS Packages on SQL Server 7.0 or Earlier**

DTS packages created on an instance of SQL Server 2000 cannot be loaded or run on an instance of SQL Server version 7.0 or earlier. If you attempt to do this, you may receive one of the following messages:

• "Invalid class string."

• "Parameter is incorrect."

Both messages indicate that the current server does not contain all the components necessary to load the package and cannot support objects defined in the DTS package, such as tasks and transformations.

However, if you receive one of these messages, you can still open and run the package on an instance of SQL Server 2000.

# Using DTS with Different Collations, Different Code Pages, and Non-Unicode Data

When using the Copy SQL Server Objects task and Copy Column transformation to copy non-Unicode data between an instance of SQL Server 2000 and SQL Server 7.0, issues arise when using different code pages and collations. For more information, see <u>Data Conversion and Transformation</u> <u>Considerations</u>.

## **Specifying Trusted Connections**

In SQL Server 7.0, you did not have to code "trusted\_connection=yes" in your connection strings for ADO, OLE DB, or ODBC to obtain a trusted connection. If you did not specify a UID and PASSWORD, SQL Server would default to trying a trusted connection. In SQL Server 2000, you must code "trusted\_connection=yes" to obtain trusted connection.

# **Extended Objects in SQL-DMO**

Some objects in SQL-DMO are extended in SQL Server 2000. For more information about using extended SQL-DMO objects, methods, and properties with SQL Server 7.0 or earlier, see <u>Programming Extended SQL-DMO Objects</u>.

## SQL-SCM

The SQL-SCM (Service Control Manager) API has been removed and is no longer supported.

### **English Query and OLAP Services for SQL Server 7.0**

For users of OLAP Services for SQL Server 7.0 who want to install or uninstall English Query, these issues apply:

- OLAP Services for SQL Server 7.0 must not be running during installation. Shut down the OLAP Services service before installing English Query. (See the Services application in Control Panel.)
- If you have installed OLAP Services for SQL Server 7.0 and you uninstall English Query, you must reinstall OLAP Services. Conversely, if you have installed English Query and you uninstall OLAP Services, you must reinstall English Query to maintain OLAP connectivity.

These issues do not occur with SQL Server 2000 Analysis Services (formerly OLAP Services).

# **SQL Server 2000 and SQL Server version 6.5**

Microsoft® SQL Server<sup>™</sup> 2000 is compatible with SQL Server version 6.5 in many respects. Most product functionality of SQL Server version 6.5 remains in SQL Server 2000. Most applications for SQL Server 6.5 work unchanged after the database server is upgraded to SQL Server 2000 by the SQL Server Upgrade Wizard.

The SQL Server 2000 upgrade process:

- Adds functionality, either new to SQL Server 2000 or changed from earlier versions, which makes tasks easier to accomplish.
- Minimizes the time and effort needed to upgrade.

In some cases, compatibility issues can arise:

• Configuration Options

Some server configuration options have changed.

• SQL-DMO, Tasks, and Replication

Task, replication, and device objects have changed. SQL Server 2000 uses jobs instead of tasks, and provides new system tables and system stored procedures.

• Replication and Triggers

Replication types that allow data modifications at the Subscriber use triggers to track changes to published tables. If there are triggers on your application that modify published tables, the **sp\_configure** server option **nested triggers** should be enabled. This option affects tables used in merge replication or tables used in snapshot replication or transactional replication with the immediate updating or queued updating option. Before adding these types of replication to an existing database that uses triggers, be sure your application works correctly with the **nested triggers** option enabled. The **nested triggers** option is enabled by default; however, if this option was disabled previously, you will need to enable it again.

• Segments and Devices

SQL Server 7.0 and SQL Server 2000 use files and filegroups instead of segments and devices for storing indexes or tables. Unless your application depends upon the physical layout of segments within devices, this does not create compatibility problems for your application.

• System Tables

If your applications depend upon accessing system tables directly, the applications may need to be revised. It is recommended that you use system stored procedures or information schema views.

Here are the SQL Server 6.*x* system tables that are not included with SQL Server 2000.

master.dbo.spt_datatype_info	sysprocedures
sysbackupdetail	sysrestoredetail
sysbackuphistory	sysrestorehistory
syshistory	syssegments
syskeys	systasks
syslocks	sysusages

• Backup and Restore

SQL Server 2000 uses BACKUP and RESTORE statements in place of DUMP and LOAD. DUMP and LOAD are supported for backward compatibility, but with some limitations.

• System Stored Procedures

Some system stored procedures are no longer supported.

For more information, see the discussion of specific backward compatibility issues.

#### Setting a Backward Compatibility Level

When running at its default settings, Microsoft SQL Server 2000 implements SQL-92 behaviors for some Transact-SQL statements whose behaviors differed from the standard in earlier versions of SQL Server. SQL Server 2000 also enforces reserved keywords that were not keywords in earlier versions of SQL Server. If upgrading existing systems with existing applications, you can use the database compatibility level settings to retain the earlier behaviors if your existing applications depend on those behaviors. This gives you time to upgrade applications in an orderly fashion. Most applications, however, are not affected by the changes in behavior and work at the SQL Server 2000 compatibility level.

The compatibility level is specified for each database using the **sp\_dbcmptlevel** system stored procedure. The database compatibility level can be set to 60 (version 6.0 compatibility), 65 (version 6.5 compatibility), 70 (version 7.0 compatibility), and the default 80 (SQL Server 2000 compatibility). The effects of the compatibility level settings are generally limited to the behaviors of a small number of Transact-SQL statements that also existed in earlier versions of SQL Server. Even when the database compatibility level is set to 60 or 65, applications gain almost all of the benefits of the new performance enhancements of SQL Server 2000. Applications still benefit from features such as the improved query processor. For more information, see the discussion of specific behaviors controlled by the different settings in <u>sp\_dbcmptlevel</u>.

For installations of all instances of SQL Server 2000, the default level for all databases is 80. For upgrades from SQL Server 7.0 to SQL Server 2000, the default level for all databases is 80. For upgrades from SQL Server 6.5 and SQL Server 6.0 to SQL Server 2000, the existing default compatibility level is retained.

**IMPORTANT** The compatibility level for the **master** database is 80 and cannot be changed. If you have added any user-defined objects to **master**, you must ensure they work correctly at the 80 compatibility level.

The **model** database is set automatically to the SQL Server 2000 compatibility level during an upgrade. All new user-defined databases are created with the same compatibility level setting as **model**. If you do not want to use any SQL Server 2000 behavior in new databases created after an upgrade, use **sp\_dbcmptlevel** to change the compatibility level setting in **model**.

Certain behaviors are not enabled at lower compatibility levels. For example, the keywords LEFT, OUTER, and JOIN are not keywords at compatibility level 60.

This means the database compatibility level must be set to 65 or higher before the LEFT OUTER JOIN clause becomes valid. Before any applications can take advantage of features only available at a higher compatibility level, all applications using the database must be upgraded to work correctly at the higher compatibility level.

Likewise, setting the compatibility level of a database to 65 makes the database version-6.5 compatible, but does not necessarily provide version 6.5 behaviors. For example, when SET ANSI\_PADDING is ON and you attempt to insert the strings 'abc' and 'abc ' into a primary key column, SQL Server 2000 considers the strings to be duplicates and does not violate the primary key constraint. In SQL Server 6.5, the two strings are considered to be unique and both insertions succeed. Setting the compatibility level to 65 does not force SQL Server 2000 to treat the strings as unique values.

**Note** While running at compatibility level 60 or 65 preserves legacy behaviors on SQL Server 2000, support for these behaviors may be dropped in future versions of SQL Server. It is recommended that you plan to upgrade your applications to work correctly with the compatibility level set to 80 as soon as is practicable.

#### See Also

Reserved Keywords System Stored Procedures System Tables

# **SQL Server Backward Compatibility Details**

Microsoft® SQL Server<sup>™</sup> 2000 adds many new features. Most of the changes are internal and will not affect your database scripts or applications. All Transact-SQL statements are compatible. However, administration tools or scripts should be updated to work with SQL Server 2000.

The backward compatibility topics in this section contain a detailed list of features and behaviors supported in SQL Server version 6.5 that have changed and could possibly affect your administration tools or scripts. These changes are not controlled by the backward compatibility level.

To indicate their potential effect on administration tools or scripts, feature changes have been grouped into four levels.

Level	Consists of
1	Administrative statements, stored procedures, or SQL Server items that have been removed from, or are no longer supported in, SQL Server 2000. Administrative tools or scripts using these items must be fixed prior to using SQL Server 2000. For more information about these features, see Level 1: Handling Discontinued Functionality.
2	Important changes that produce different behavior from earlier versions of SQL Server. For example, items in this category are those that have changed behavior in data type conversion or usage of selected functions, changed behavior of clauses in selected Transact-SQL statements and stored procedures, changed column names in selected system tables, and changed behavior due to the database compatibility setting. For more information about these features, see Level 2: Handling Major Changes to Behavior.
3	Items supported for backward compatibility only. Any item included in this category is fully supported, but may be removed or unsupported in a future release. SQL Server 2000 provides features that accomplish these tasks more efficiently and have ongoing support. For more information about these features, see <u>Level 3</u> :

	Updating to Improve Earlier Functionality.	
4	Minor changes that produce different behavior from earlier	
	versions of SQL Server. For example, items in this category are	
	either ignored or have one or more ignored parameters, changed	
	byte lengths, added parameters or columns, or changed data type	
	columns. For more information about these features, see Level 4:	
	Handling Minor Changes to Behavior.	

**Note** You might find it helpful to review <u>SQL Server 2000 and SQL Server</u> <u>version 7.0</u> as well.

# **SetHostName property not used in SQL Server 2000**

When using SQL Server 6.5 integrated security, SQL Server 6.5 did not report the Windows NT account used by a connection unless the system administrator activated the SET HOSTNAME TO USERNAME option in SQL Enterprise Manager.

The setting could also be activated through the SQL-DMO SetHostName property. With this setting in effect, these functions and columns returned the user's Windows NT account name instead of the network name of the client computer:

- Transact-SQL HOST\_NAME() function
- **hostname** column in the result set returned by **sp\_who**
- hostname column in sysprocesses

In SQL Server 2000, the **loginame** column in the **sp\_who** result set contains the Windows NT account name for connections made using Windows NT Authentication. Applications needing the Windows NT account associated with a connection using Windows NT Authentication should reference this column.

SQL Server 2000 Enterprise Manager no longer presents the SET HOSTNAME TO USERNAME option. SQL Server 2000 ignores the setting of the SQL-DMO SetHostName property.

# **Level 1: Handling Discontinued Functionality**

Backward Compatibility Level 1 consists of administrative statements, stored procedures, or Microsoft® SQL Server<sup>™</sup> items that were supported in SQL Server 6.5 but have been removed from, or are no longer supported in, SQL Server 2000. Administrative tools or scripts using these items must be fixed prior to using SQL Server 2000.

This subheading	Relates to
Backup and Restore	BACKUP
	RESTORE
	DUMP
	LOAD
	sysbackuphistory
	sysbackupdetail
	sysrestorehistory
	sysrestoredetail
	backupfile
	backupmediafamily
	backupmediaset
	backupset
	restorefile
	restorefilegroup
	restorehistory
Configuration Options	<pre>sp_configure (backup buffer size, backup</pre>
	threads, database size, free buffers, hash
	buckets, LE threshold maximum, LE
	threshold minimum, LE threshold
	percent, logwrite sleep, max lazywrite IO,
	memory, open databases, procedure
	cache, RA cache hit limit, RA cache miss
	limit, RA delay, RA pre-fetches, RA slots
	per thread, RA worker threads, recovery
	flags, remote conn timeout, SMP

	concurrency, sort pages, min memory per query, index create memory, tempdb in ram, and user connections options) trace flag 204
<u>Custom Sort Orders</u>	Character sets, sort orders, and Unicode collations
<u>Databases</u>	ALTER DATABASE
Database Options	<pre>sp_dboption (subscribe and no chkpt. on</pre>
	recovery options)
	sp_addsubscription
	RESTORE
Data Access Objects (DAO)	odbccmpt utility
DBCC	DBCC DBREINDEX
	DBCC MEMUSAGE
	DBCC SHRINKDB
DB-Library	Two-Phase Commit
	DB-Library for Visual Basic
DECnet Network Library	DECnet Sockets Net-Library
Disk Commands	DISK REINIT
	DISK REFIT
	ALTER DATABASE
Disk Mirroring	DISK MIRROR
	DISK REMIRROR
	DISK UNMIRROR
<u>Indexes</u>	CREATE INDEX
<u>Open Data Services</u>	Windows NT Component Services
	SRV_CONFIG
	SRV_PROC
	SRV_SERVER
	srv.h
	Opends60.lib
Program Group Tools and	Client Network Utility
<u>Utilities</u>	ISQL_w
	MS Query
	SQL Client Configuration

	SQL Enterprise Manager
	SQL Help
	SQL Security Manager
	SQL Trace
	SQL Performance Monitor
	SQL Service Manager
	SQL Setup
	SQL Query Analyzer
	SQL Server Enterprise Manager
	SQL Server Profiler
	SQL Server Service Manager
Replication	Restricted publications
-	<b>DBOption</b> object
	ReplicationDatabase object
	EnablePublishing property
	<b>repl_publisher</b> login
Security	DENY
	Delimited Identifiers
<u>Segments</u>	CREATE INDEX
	CREATE TABLE
	sp_addsegment
	sp_dropsegment
	sp_extendsegment
	sp_helpsegment
	CREATE DATABASE
	ALTER DATABASE
Services	SQL Executive
SET	SET DISABLE_DEF_CNST_CHK
DISABLE_DEF_CNST_CHK	
SET SHOWPLAN	SET SHOWPLAN
	SET SHOWPLAN_ALL
	SET SHOWPLAN_TEXT
SQL Alerter	SQLALRTR.exe
<u>SQL-DMO</u>	sqlole.dll
System Stored Procedures	xp_snmp_getstate
(General Extended	xp_snmp_raisetrap

Procedures)	
System Stored Procedures	sp_replica
(Replication)	sp_replsync
	sp_helppublicationsync
	sp_subscribe
	sp_unsubscribe
	@@ERROR
	sp_changepublication
	sp_addpublisher
	sp_adddistpublisher
	sp_droppublisher
	sp_dropdistpublisher
	sp_distcounters
	sp_helpreplicationdb
	sp_helpreplicationdboption
	sp_replstatus
System Stored Procedures	ALTER TABLE
(System)	CREATE TABLE
	sp_help
	sp_helpconstraint
	sp_commonkey
	sp_dropkey
	sp_foreignkey
	sp_helpjoins
	sp_helpkey
	sp_primarykey
	sp_placeobject
	sp_dbinstall
	sp_attach_db
	sp_makestartup
	sp_unmakestartup
	sp_procoption
	sp_helplogins
	sp_helprotect
	sp_tableoption
	<pre>sp_serveroption (fallback option) </pre>
	sp_setlangalias

	<pre>sp_droplanguage sp_fallback_activate_svr_db sp_fallback_deactivate_svr_db sp_fallback_enroll_svr_db sp_fallback_help sp_fallback_permanent_svr sp_fallback_upd_dev_drive sp_fallback_upd_dev_drive sp_fallback_withdraw_svr_db sp_devoption sp_diskdefault sp_helplog sp_helpstartup sp_help_revdatabase sp_sqlexec sp_addlanguage</pre>
System Stored Procedures (Tasks)	sp_addalert sp_addnotification
	sp_addoperator
	sp_dropalert
	sp_dropnotification
	sp_dropoperator
	sp_helpalert
	sp_helphistory
	sp_helpnotification
	sp_helpoperator
	sp_purgehistory
	sp_runtask
	sp_stoptask
	sp_updatealert
	sp_updatenotification
	sp_updateoperator
	sp_add_alert
	sp_add_notification
	sp_add_operator
	sp_delete_alert
	sp_delete_notification

	sp_delete_operator sp_help_alert sp_help_jobhistory sp_help_notification sp_help_operator sp_purge_jobhistory sp_start_job sp_stop_job sp_update_alert sp_update_notification sp_update_operator
<u>System Tables</u>	Information Schema Views System Stored Procedures (Catalog Procedures) sysdevices (mirrorname and stripeset columns) syshistory sysjobhistory sysjobhistory sysindexes (distribution, segment, rowpage, keys1, and keys2 columns) syskeys syslocks syslocks syslockinfo syslogs sysprocedures sysprocedures syscomments syscomments syscomments CREATE DATABASE ALTER DATABASE ALTER TABLE ALTER TABLE CREATE TABLE ALTER TABLE CREATE INDEX systasks sysjobs sysjobsteps sysjobsteps sysjobservers

	sysusages master.dbo.spt_datatype_info
<u>Transactions</u>	Data type conversions
<u>Utilities</u>	probe login

## Level 2: Handling Major Changes to Behavior

Backward Compatibility Level 2 consists of important changes in Microsoft® SQL Server<sup>™</sup> 2000 that produce different behavior from earlier versions of SQL Server. For example, items in this category are those that have changed behavior in data type conversion or usage of selected functions, changed behavior of clauses in selected Transact-SQL statements and stored procedures, changed column names in selected system tables, and changed behavior due to the database compatibility setting. This topic covers backward compatibility details for these items.

This subheading	Relates to
Backup and Restore	BACKUP
	CREATE DATABASE
	ALTER DATABASE
	RESTORE
	sp_dboption
<u>Bulk Copy</u>	bcp Utility
Configuration Options	Setting Configuration Options
	<pre>sp_configure (open objects and user</pre>
	connections options)
Database Pages and Extents	Pages and Extents
<u>Data Types</u>	CAST and CONVERT
	Data Types
DB-Library	dbcursorfetchex
Empty Strings	sp_dbcmptlevel
	CHARINDEX
	DATALENGTH
	LEFT
	LTRIM
	PATINDEX
	REPLICATE
	RIGHT
	RTRIM

SPACE SUBSTRING UPDATETEXTIndexesCREATE INDEXINSERT <b>sp_dbcmptlevel</b> INSERTKeyset CursorsKeyset cursorsLTRIM and RTRIM Trimming FunctionsLTRIM RTRIMODBCSQLGetDiagRec SQLMoreResultsRIGHTUsing Identifiers Reserved Keywords	
UPDATETEXTIndexesCREATE INDEXINSERT <b>sp_dbcmptlevel</b> INSERTKeyset CursorsKeyset cursorsLTRIM and RTRIMLTRIM RTRIMTrimming FunctionsRTRIMODBCSQLGetDiagRec SQLMoreResultsRIGHTUsing Identifiers	
INSERTsp_dbcmptlevel INSERTKeyset CursorsKeyset cursorsLTRIM and RTRIMLTRIM RTRIMTrimming FunctionsRTRIMODBCSQLGetDiagRec SQLMoreResultsRIGHTUsing Identifiers	
INSERTKeyset CursorsKeyset cursorsLTRIM and RTRIMLTRIMTrimming FunctionsRTRIMODBCSQLGetDiagRec SQLMoreResultsRIGHTUsing Identifiers	
LTRIM and RTRIM       LTRIM         Trimming Functions       RTRIM         ODBC       SQLGetDiagRec         SQLMoreResults       SQLMoreResults         RIGHT       Using Identifiers	
Trimming FunctionsRTRIMODBCSQLGetDiagRec SQLMoreResultsRIGHTUsing Identifiers	
ODBC     SQLGetDiagRec       SQLMoreResults       RIGHT       Using Identifiers	
SQLMoreResults       RIGHT     Using Identifiers	
C C	
Reserved Keywords	
Security GRANT	
REVOKE	
DENY	
sp_addlinkedsrvlogin	
<u>SELECT</u> SELECT	
SET SHOWPLAN SET SHOWPLAN_ALL	
SET SHOWPLAN_TEXT	
System TablesInformation Schema Views	
System Stored Procedures (Cata	log
Procedures)	_
sysdatabases (logptr and dump	trdate
columns)	
sysmessages (langid column)	
<b>syslogins (language</b> column) computed columns	
Table Hints DELETE FROM	
INSERT	
SELECT	
UPDATE	
Transactions SET TRANSACTION ISOLATI	ON LEVEL
SET CURSOR_CLOSE_ON_CO	

	ROLLBACK
	DECLARE CURSOR
Triggers and System Stored	sp_dbcmptlevel
Procedures	sp_create_removable
	CREATE TRIGGER
	SET QUOTED_IDENTIFIER
	SET ANSI_NULLS
	SET ANSI_DEFAULTS
<u>UPDATE</u>	@@ERROR
	UPDATE
	INSERT
<u>UPDATETEXT</u>	UPDATETEXT
	WRITETEXT
<u>Views</u>	DELETE
	INSERT
	UPDATE

## **Level 3: Updating to Improve Earlier Functionality**

Backward Compatibility Level 3 consists of items that were supported in SQL Server version 6.5 but are supported in SQL Server 2000 (and SQL Server 7.0) for backward compatibility only. Any item included in this category is fully supported, but may be removed or unsupported in a future release. It is recommended that, as time allows, the backward compatible item be replaced with the recommended item. SQL Server 2000 provides features that accomplish these tasks more efficiently and have ongoing support.

This subheading	Relates to
Backup and Restore	BACKUP
	RESTORE
	CREATE DATABASE
Database Options	<pre>sp_dboption (publish option)</pre>
	sp_replicationdboption
DBCC	DBCC NEWALLOC
	DBCC CHECKALLOC
	DBCC ROWLOCK
	Architecture Enhancements
	DBCC TEXTALL
	DBCC CHECKDB
	DBCC TEXTALLOC
	DBCC CHECKTABLE
	DBCC DBREPAIR
	DROP DATABASE
Devices	Overview of SQL Server Architecture
	DISK INIT
	CREATE DATABASE
	ALTER DATABASE
	DISK REINIT
	sp_logdevice
	sp_dropdevice

This topic covers backward compatibility details for these items.

srv_paramdata srv_paramlen
-------------------------------

	-
	srv_parammaxlen
	srv_paramname
	srv_paramnnumber
	srv_paramset
	srv_paramstatus
	srv_paramtype
	srv_returnval
	srv_rpcdb
	srv_rpcnumber
	srv_rpcoptions
	srv_clearstatistics
	srv_sendstatistics
	srv_alloc
	srv_bmove
	srv_bzero
	srv_free
	srv.h
	srv_describe
	srv_setcollen
	srv_setcoldata
	srv_paramsetoutput
	srv_paraminfo
Query Performance	SUSER_ID
	SUSER_SID
	SUSER_NAME
	SUSER_SNAME
	syslogins
	sysdatabases
	sysremotelogins
	sysusers
	sysalternates
<u>Security</u>	GRANT
	Authentication
	SETUSER
SELECT	FASTFIRSTROW
<u>SELECT</u>	FASTFIRSTROW SELECT
<u>SELECT</u>	

<u>SET SHOWPLAN</u>	SET SHOWPLAN_TEXT SET SHOWPLAN_ALL <b>SQLGetDiagRec</b>
System Stored Procedures	xp_grantlogin
(Extended)	xp_revokelogin
	sp_grantlogin
	sp_revokelogin
System Stored Procedures (System)	sp_add_job
	sp_add_jobschedule
	sp_add_jobstep
	sp_addtask
	sp_delete_job
	sp_delete_jobschedule
	sp_delete_jobstep
	sp_droptask
	sp_help_jobhistory
	sp_help_jobschedule
	sp_help_jobstep
	sp_helptask
	sp_purge_jobhistory
	sp_reassigntask
	sp_start_job
	sp_stop_job
	sp_update_job
	sp_update_jobschedule
	sp_update_jobstep
	sp_updatetask

## **Level 4: Handling Minor Changes to Behavior**

Backward Compatibility Level 4 consists of minor changes in Microsoft® SQL Server<sup>TM</sup> 2000 that produce different behavior from earlier versions of SQL Server. For example, items in this level are either ignored or have one or more ignored parameters, changes to byte lengths, added parameters or columns, or changed data type columns.

This subheading	Relates to these items
<u>Aliases</u>	Roles
	Managing Permissions
Backup and Restore	RESTORE HEADERONLY
	LOAD HEADERONLY
<b>Configuration</b>	<pre>sp_configure (media retention option)</pre>
	Setting Configuration Options
CREATE PROCEDURE	CREATE TABLE
	SELECT INTO
<u>Data Types</u>	decimal and numeric
	Using Mathematical Functions
	+ (Add)
	- (Subtract)
	* (Multiply)
	/ (Divide)
	ATN2
	AVG
	CAST and CONVERT
	EXP
	POWER
	RADIANS
	ROUND
	SUM
DATEPART and SET	SET DATEFIRST

This topic covers backward compatibility details for these items.

<u>DATEFIRST</u>	DATEPART
DBCC	DBCC
DBCS String Comparisons	Unicode space characters
DELETE and SELECT	FROM
<u>Devices</u>	ALTER DATABASE
Functions	@@DBTS
<u>Global Variables</u>	Functions
<u>ODBC</u>	SQL_COPT_SS_PERF_QUERY_INTERVAL
	SQLMoreResults
	SQL_NO_DATA
<u>Rebuilding the master</u>	Rebuild Master Utility
<u>Database</u>	
Rebuilding the Registry	setup/t RegistryRebuild = On
<u>(Level 4)</u>	
Replication	Replication Between Different Versions of
	SQL Server
	Subscribing to One or More Articles of a
	Publication
<u>Security</u>	SYSTEM_USER
<u>SELECT</u>	SELECT
	FROM
Triggers and System Stored	CREATE TRIGGER
Procedures (System)	<pre>sp_dboption (recursive triggers option)</pre>
	sp_tableoption
	xp_readmail
	xp_sendamil
<u>UPDATE</u>	UPDATE
<u>Utilities</u>	SQL Query Analyzer
	isql utility

## How to Install SQL Server 2000

This set of How To topics includes common procedures used in installing Microsoft® SQL Server<sup>™</sup> 2000.

### How to install SQL Server 2000 (Setup)

#### To install SQL Server 2000

- 1. Insert the Microsoft® SQL Server<sup>™</sup> 2000 compact disc in your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select SQL Server 2000 Components.

If you are running Microsoft Windows® 95, click **SQL Server 2000 Prerequisites**, and then click **Install Common Controls Library Update.** 

- 3. Select **Install Database Server** and setup prepares the SQL Server Installation Wizard. At the **Welcome** screen, click **Next**.
- 4. In the **Computer Name** dialog box, **Local Computer** is the default option and the local computer name appears in the edit box. Click **Next**.

For a remote installation, click **Remote Computer**. You can then type a computer name or click **Browse** to locate a remote computer.

If a cluster is detected, **Virtual server** is the default option.

- 5. In the **Installation Selection** dialog box, click **Create a new instance of SQL Server, or install Client Tools**, and then click **Next**.
- 6. Follow directions on the **User Information**, **Software License Agreement** and related screens.
- 7. In the **Installation Definition** dialog box, click **Server and Client Tools**, and then click **Next**.
- 8. In the **Instance Name** dialog box, if the **Default** check box is

available, you can install either the default or a named instance. If the **Default** check box is not available, a default instance has already been installed, and you can install only a named instance.

- To install the default instance, select the **Default** check box, and click **Next**.
- To install a named instance, clear the **Default** check box, and type a new named instance in the **Instance Name** edit box. Click **Next**.
- 9. In the **Setup Type** dialog box, click **Typical** or **Minimum**, and then click **Next**.

If you want to select components and subcomponents, change character set, network libraries or other settings, click **Custom**, and then click **Next**.

10. In the **Service Accounts** dialog box, accept the default settings, enter your domain password, and then click **Next**.

For information about services account options, see <u>Services</u> <u>Accounts</u>.

11. In the **Authentication Mode** dialog box, accept the default setting, and click **Next**.

To use Mixed Mode, see <u>Authentication Modes</u>.

- 12. When you are finished specifying options, click **Next** in the **Start Copying Files** dialog box.
- 13. In the **Choose Licensing Mode** dialog box, make selections according to your license agreement, and click **Continue** to begin the installation.

Click **Help** for information about licensing or see your system administrator.

14. In the Setup Complete dialog box, click Yes, I want to restart my

**computer now**, and then click **Finish**.

### See Also

- How to add components to an instance of SQL Server 2000 (Setup)
- How to create a case-sensitive instance of SQL Server (Setup)

How to install a named instance of SQL Server (Setup)

## How to install client tools only (Setup)

You can install client tools only using any SQL Server compact disc, on any supported operating system. For more information, see <u>How to install tools only</u> <u>from any compact disc</u>.

#### To install client tools only for SQL Server 2000

- 1. Insert the Microsoft® SQL Server<sup>™</sup> 2000 compact disc in your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select **SQL Server 2000 Components**, select **Install Database Server**, and then click **Next** at the Welcome screen of the SQL Server Installation Wizard.
- 3. In **Computer Name** dialog box, **Local Computer** is the default option, and the local computer name appears in the edit box. Click **Next**.
- 4. In the **Installation Selection** dialog box, click **Create a new instance of SQL Server, or install Client Tools**, and then click **Next**.
- 5. Follow the directions on the **User Information**, **Software License Agreement**, and related screens.
- 6. In the **Installation Definition** dialog box, click **Client tools only**, and then click **Next**.
- 7. In the **Select Components** dialog box, accept the defaults or select the components you want, and then click **Next**.

You can select an item in the **Components** list, such as **Management Tools**, and then select items from the related **Sub-Components** list, such as **Enterprise Manager**. Click to select items you want to install; clear the check box of the items you do not want to install.

For information about each component, select the item, and view the Description box.

8. In the **Start Copying Files** dialog box, click **Next** to complete the installation of the client tools.

## How to install tools only from any compact disc (Setup)

**Note** In this procedure, you can use the installation disc for any edition of SQL Server 2000 on a computer with any of the operating systems supported by SQL Server 2000.

#### To install tools only from any compact disc

- 1. Insert a Microsoft® SQL Server<sup>™</sup> 2000 compact disc in your CD-ROM drive. This can be the installation disc for any edition of SQL Server 2000, without regard to operating system support. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select **SQL Server 2000 Components**, select **Install Database Server**, and then click **Next** at the Welcome screen of the SQL Server Installation Wizard.
- 3. In **Computer Name** dialog box, **Local Computer** is the default option, and the local computer name appears in the edit box. Click **Next**.
- 4. Follow the directions on the **User Information**, **Software License Agreement**, and related screens.
- 5. In the **Select Components** dialog box, accept the defaults or select the components you want, and then click **Next**.

You can select an item in the **Components** list, such as **Management Tools**, and then select items from the related **Sub-Components** list, such as **Enterprise Manager**. Click to select items you want to install; clear the check box of the items you do not want to install.

For information about each component, select the item, and view the

Description box.

6. In the **Start Copying Files** dialog box, click **Next** to complete the installation of the client tools.

## How to install connectivity only (Setup)

The connectivity-only option installs Network Libraries and MDAC (Microsoft® Data Access Components).

#### To install connectivity only for SQL Server 2000

- 1. Insert the Microsoft SQL Server<sup>™</sup> 2000 compact disc in your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select SQL Server 2000 Components.

If you are running Microsoft Windows® 95, click **Install Common Controls Library Update.** 

- 3. Select **Install Database Server** and setup prepares the SQL Server Installation Wizard. At the **Welcome** screen, click **Next**.
- 4. In the **Computer Name** dialog box, **Local Computer** is the default option, and the local computer name appears in the edit box. Click **Next**.
- 5. In the **Installation Selection** dialog box, click **Create a new instance of SQL Server, or install Client Tools**, and then click **Next**.
- 6. Follow the directions on the **User Information, Software License Agreement** and related screens.
- 7. In the **Installation Definition** dialog box, click **Connectivity Only**, and then click **Next**.
- 8. In the **Start Copying Files** dialog box, click **Next** to complete the installation.

# How to install a named instance of SQL Server 2000 (Setup)

You can install a named instance of Microsoft® SQL Server<sup>™</sup> 2000 the first time you run SQL Server Setup or later after the default instance is installed. For each additional named instance you want to install, follow this procedure.

**Note** If you have a SQL Server 7.0 installation on your computer, the installation remains intact during the installation of a named instance of SQL Server 2000. A default instance of SQL Server 2000 will overwrite a SQL Server 7.0 installation (as the previous default installation), but a named instance does not overwrite SQL Server 7.0.

#### To install a named instance of SQL Server 2000

- 1. Insert the SQL Server 2000 compact disc in your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select SQL Server 2000 Components.

If you are running Microsoft Windows® 95, **Install Common Controls Library Update.** 

- 3. Select **Install Database Server** and setup prepares the SQL Server Installation Wizard. At the **Welcome** screen, click **Next**. In the **Computer Name** dialog box, **Local Computer** is the default option, and the local computer name appears in the edit box. Click **Next**.
- 4. In the **Installation Selection** dialog box, click **Create a new instance of SQL Server, or install Client Tools,** and then click **Next**.
  - If this is the first SQL Server 2000 installation on your computer, follow the directions on the **User Information**, **Software License Agreement**, and related screens.

- If an installation of SQL Server 2000 exists on your computer, these screens are omitted.
- 5. In the **Installation Definition** dialog box, click **Server and Client Tools**, and then click **Next**.
- 6. In the **Instance Name** dialog box, clear the **Default** check box, and type a name for the new named instance, and then click **Next**.

**Note** If you have an existing default installation (either SQL Server 7.0 or 2000), the Default check box is not available.

If you have typed an instance name, and later return to the Instance Name dialog box to change the name before completing setup, you can do so. However, a workaround is necessary to edit the instance name box, which will be unavailable after clicking Back to get to this dialog box. Select the Default checkbox, then immediately clear it, and you will be able to edit the instance name.

For more information about instance names, click Help.

7. In the **Setup Type** dialog box, select **Typical**, **Minimum**, or **Custom**, and then click **Next**.

If you want to select subcomponents or change character set, network libraries, or other settings, click **Custom**.

8. In the **Service Accounts** dialog box, accept the default settings, enter your domain password, and then click **Next**.

For information about services account options, see <u>Services</u> <u>Accounts</u>.

9. In the **Authentication Mode** dialog box, accept the default setting, and click **Next**.

To use **Mixed Mode** authentication, see <u>Authentication Modes</u>.

10. When you are finished specifying options, click **Next** in the **Start Copying Files** dialog box.

11. In the **Choose Licensing Mode** dialog box, make selections according to your license agreement, and click **Continue** to begin the installation.

Click **Help** for information about licensing, or see your system administrator.

12. In the **Setup Complete** dialog box, click **Yes, I want to restart my computer now**, and then click **Finish**.

#### See Also

Working with Named and Multiple Instances of SQL Server 2000

Running SQL Server 7.0 Along with a Named Instance of SQL Server 2000

## How to upgrade a SQL Server 7.0 installation to SQL Server 2000 (Setup)

**CAUTION** This version upgrade procedure overwrites your Microsoft® SQL Server<sup>TM</sup> 7.0 installation; the installation no longer exists on your computer. In addition, previous registry settings are removed. For example, after upgrading you will need to re-register your servers.

To restore the SQL Server 7.0 installation, you must first uninstall SQL Server 2000, perform a complete reinstall of the SQL Server 7.0 files, and then restore your backed-up SQL Server 7.0 databases.

#### To upgrade SQL Server 7.0 to SQL Server 2000

1. Insert the Microsoft SQL Server 2000 compact disc for the edition to which you want to upgrade into your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.

**Note** If you have purchased an edition of SQL Server with more features than your current SQL Server 7.0 installation, the upgrade process will perform both the version and edition upgrade at the same time.

- 2. Select **SQL Server 2000 Components**, select **Install Database Server**, and then setup prepares the SQL Server Installation Wizard. At the **Welcome** screen, click **Next**.
- 3. In **Computer Name** dialog box, **Local Computer** is the default option and the local computer name appears in the edit box. Click **Next**.
- 4. In the **Installation Selection** dialog box, click **Upgrade, remove, or add components to an existing instance of SQL Server**, and then click **Next**.
- 5. In the **Instance Name** dialog box, **Default** will be selected. Click

#### Next.

**Note** When upgrading, SQL Server 7.0 automatically becomes the default instance of SQL Server 2000.

- 6. In the **Existing Installation** dialog box, click **Upgrade your existing installation**, and then click **Next**.
- 7. In the Upgrade dialog box, you are prompted as to whether you want to proceed with the requested upgrade. Click Yes, upgrade my <text specific to the upgrade> to start the upgrade process, and then click Next. The upgrade runs until finished.
- 8. In the **Connect to Server** dialog box, select an authentication mode, and then click **Next**.

If you are not sure which mode to use, accept the default: **The Windows account information I use to log on to my computer with** (Windows).

- 9. In Start Copying Files dialog box, click Next.
- 10. In the **Setup Complete** dialog box, click **Yes, I want to restart my computer now**, and then click **Finish**.

#### See Also

Authentication Modes

How to perform an edition upgrade within SQL Server 2000 (Setup)

# How to upgrade databases online using the Copy Database Wizard (Enterprise Manager)

To upgrade a SQL Server 7.0 database to a SQL Server 2000 database

- 1. Expand a server group, and then expand a server.
- 2. Right-click the server, point to **All Tasks**, and then click **Copy Database Wizard**.
- 3. Complete the steps in the wizard.

**IMPORTANT** After upgrading databases from SQL Server 7.0, run sp\_updatestats (update statistics) against the database on the destination server to ensure optimal performance of the copied database.

## See Also

Copy Database Wizard Help

Database Upgrade from SQL Server 7.0 (Copy Database Wizard)

# How to perform an edition upgrade within SQL Server 2000 (Setup)

# To upgrade a SQL Server 2000 installation to a different edition of SQL Server 2000

- 1. Insert the Microsoft® SQL Server 2000<sup>™</sup> compact disc for the edition you want to install into your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select **SQL Server 2000 Components**, select **Install Database Server**, and then setup prepares the SQL Server Installation Wizard. At the **Welcome** screen, click **Next**.
- 3. In **Computer Name** dialog box, select **Local Computer** or **Remote computer**.
- 4. In the **Installation Selection** dialog box, click **Upgrade**, **Remove**, **or Add Components to an existing instance of SQL Server**, and then click **Next**.
- 5. In the **Instance Name** dialog box, click **Next**.
- 6. In the **Existing Installation** dialog box, click **Upgrade your existing installation**, and then click **Next**.
- If Setup detects that you are doing an edition upgrade, the Upgrade dialog box appears. Click Yes, Upgrade my <text specific to the upgrade> to upgrade the feature set of your current installation, and click Next.

8. After the upgrade is completed, you are prompted as to whether you want to install additional components. If you click **Yes**, the **Select Components** dialog box appears. Accept the defaults or select the additional components you want to install, and then click **Next**.

You can select an item in the **Components** list, and then select items from the related **Sub-Components** list. Click to select items you want to install; clear the check box of the items you do not want to install.

- 9. When you are finished specifying options, in the **Start Copying Files** dialog box, click **Next**.
- 10. In the **Setup Complete** dialog box, click **Yes, I want to restart my computer now**, and then click **Finish** to complete the edition upgrade.

### See Also

Upgrading an Existing Installation of SQL Server

# How to uninstall an existing installation of SQL Server (Setup)

# To uninstall an existing installation of SQL Server 7.0 or SQL Server 2000 (default or named instance)

- 1. Insert the Microsoft® SQL Server<sup>™</sup> 2000 compact disc in your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select **SQL Server 2000 Components**, select **Install Database Server**, and then setup prepares the SQL Server Installation Wizard. At the **Welcome** screen, click **Next**.
- 3. In **Computer Name** dialog box, select **Local Computer** or **Remote computer**.
- 4. In the **Installation Selection** dialog box, click **Upgrade, Remove, or Add Components to an existing instance of SQL Server**, and then click **Next**.
- 5. In the **Instance Name** dialog box, **Default** is selected if you have the Default instance installed. If you want to uninstall a named instance, select it from the Instance Name list box, and then click **Next**.
- 6. In the **Existing Installation** dialog box, click **Uninstall your existing installation**, and then click **Next**.
- 7. Setup removes the selected installation. In the **Uninstalling** dialog box, click **Next**, and then in the **Setup Complete** dialog box, click **Finish**.

# How to test an installation of SQL Server 2000 (Command Prompt)

### To test the installation

1. Start Microsoft<sup>®</sup> SQL Server<sup>™</sup> 2000 by entering from a command prompt:

For the default instance, use:

net start mssqlserver

For a named instance, include the instance name, for example:

net start MSSQL\$Instance1

2. Connect to SQL Server by entering:

For the default instance, use:

osql /Usa /P <administrator password>

For a named instance, include both the server and instance name, for example:

osql /Usa /P /S Machine1\Instance1

When **osql** connects, this **osql** prompt appears:

1>

If **osql** cannot connect, an ODBC error is returned.

 Enter a simple query, such as: SELECT @@SERVERNAME GO

The **osql** utility returns the server name:

1> SELECT @@SERVERNAME 2> GO

-----

WOLFHOUND

(1 row affected) 1>

 4. Verify that you have checked a SQL Server 2000 server by entering: SELECT @@VERSION GO

The **osql** utility returns the version information.

5. Quit the **osql** utility by entering: Exit

# How to change SQL Server services login account information (Windows NT)

### To change SQL Server services login account information (Windows NT)

- 1. On the **Start** menu, point to **Settings**, and then click **Control Panel**.
- 2. Double-click **Services**.
- 3. In the **Services** dialog box, double-click **MSSQLSERVER** in the Service list.

**Note** For named instances, the instance name is included. For example, to modify the user account for **Instance1**, you double-click **MSSQL\$Instance1**.

- 4. In the **Service** dialog box, under **Log on as**, select **This account**, and then enter the changed account information.
- 5. Repeat Steps 3 and 4 above for SQL Server Agent. In the Services dialog box, double-click SQLSERVERAGENT (or SQLAgent\$Instance1 for a named instance), and then enter the changed account information in the Service dialog box.
- 6. Start SQL Server Enterprise Manager, and change the user account information there, as well, for both SQL Server and SQL Server Agent For more information see <u>How to change SQL Server services login</u> <u>account information (Enterprise Manager)</u>.

# How to change SQL Server services login account information (Windows)

To change SQL Server services login account information (Windows 2000)

- 1. On the **Start** menu, point to **Programs/Administrative Tools**, and then click **Services**.
- 2. Right-click **MSSQLServer**, and then click **Properties**.
- 3. On the **Log On** tab, enter and confirm the new password, and then restart services using the SQL Server Service Manager.
- 4. Repeat the password reset for SQLServerAgent and other services.
- 5. Start SQL Server Enterprise Manager, and change user account information there, as well, for both SQL Server and SQL Server Agent For more information, see <u>How to change SQL Server services login</u> <u>account information (Enterprise Manager)</u>.

# How to change SQL Server services login account information (Enterprise Manager)

**Note** If you are running Microsoft® Windows® 2000 and want to use the Windows 2000 Encrypted File System to encrypt any Microsoft SQL Server<sup>™</sup> files, you must unencrypt the files before you can change the SQL Server service accounts. If you do not unencrypt the files and then reset the SQL Server service accounts, you cannot unencrypt the files.

### To change the MSSQLServer service login (Enterprise Manager)

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. In the SQL Server Properties dialog box, click the **Security** tab.
- 4. In the **Startup service account** box, the option for **This Account** is selected, indicating that the SQL Server service account is a Windows domain account. Enter changes as necessary for the account and password.

### To change the SQLServerAgent service login (Enterprise Manager)

**Note** You can change the SQLServerAgent service account to a non Microsoft Windows NT® 4.0 administrator account. However, the Windows NT 4.0 account must be a member of the **sysadmin** fixed server role to run SQL Server Agent.

- 1. Expand a server group, and then expand a server.
- 2. Expand Management.
- 3. Right-click **SQL Server Agent**, and then click **Properties**.

- 4. In the **SQL Server Agent Properties** dialog box, click the **General** tab.
- 5. In the **Service startup account** box, enter the appropriate account and password.

### See Also

Creating SQL Server Services User Accounts

Changing Passwords and User Accounts

# How to rebuild the registry (Setup)

#### To rebuild the registry

- 1. Insert the Microsoft® SQL Server<sup>™</sup> 2000 compact disc in your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select **SQL Server 2000 Components**, select **Install Database Server**, and then click **Next** at the Welcome screen of the SQL Server Installation Wizard.
- 3. In the **Computer Name** dialog box, click **Next**.
- 4. In the **Installation Selection** dialog box, click **Advanced options**, and then in the **Advanced Options** dialog box, click **Registry Rebuild**. Click **Next**.
- 5. A message appears informing you that Setup rebuilds the registry based on information you supply in the subsequent screens.

**CAUTION** The setup options you enter must be the same choices that you entered during the initial installation. If you do not know or are not sure of this information, do not use this registry rebuild process. Instead, you must uninstall and reinstall SQL Server to restore the registry.

6. To prepare for the registry rebuild, enter the same information and options that you entered during the initial installation of SQL Server in the setup screens as they appear. When you have finished, the registry rebuild will occur.

**Note** Rebuilding the registry includes re-copying external components such as MDAC and MS DTC.

# How to rebuild the master database (Rebuild Master utility)

### To rebuild the master database

- 1. Shutdown Microsoft® SQL Server<sup>™</sup> 2000, and then run Rebuildm.exe. This is located in the Program Files\Microsoft SQL Server\80\Tools\Binn directory.
- 2. In the **Rebuild Master** dialog box, click **Browse**.
- 3. In the **Browse for Folder** dialog box, select the \Data folder on the SQL Server 2000 compact disc or in the shared network directory from which SQL Server 2000 was installed, and then click **OK**.
- 4. Click **Settings**. In the **Collation Settings** dialog box, verify or change settings used for the **master** database and all other databases.

Initially, the default collation settings are shown, but these may not match the collation selected during setup. You can select the same settings used during setup or select new collation settings. When done, click **OK**.

5. In the **Rebuild Master** dialog box, click **Rebuild** to start the process.

The Rebuild Master utility reinstalls the **master** database.

**Note** To continue, you may need to stop a server that is running.

## See Also

Collation Settings in Setup

# How to perform a remote installation of SQL Server 2000 (Setup)

#### To perform a remote installation

- 1. Insert the Microsoft® SQL Server<sup>™</sup> 2000 compact disc in your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select **SQL Server 2000 Components**, select **Install Database Server**, and then click **Next** at the Welcome screen of the SQL Server Installation Wizard.
- 3. In **Computer Name** dialog box, click **Remote Computer**. You can then type a computer name or click **Browse** to locate a remote computer.
- 4. In the **Installation Selection** dialog box, click **Create a new instance of SQL Server,** or **install Client Tools**.
- 5. Follow the directions on the **User Information**, Software License Agreement, and related screens.
- 6. In the **Remote Setup Information** dialog box, enter password and other information. For more information, see <u>Remote Setup</u> <u>Information</u>. After you finish defining options, click **Next**.
- 7. In the **Installation Definition**, **Instance Name**, **Setup Type**, and subsequent setup screens, select the options you want for the remote installation.

SQL Server Setup creates the Setup.iss file in your local system folder with the options you have specified.

- 8. After Setup creates Setup.iss, the **Setup Complete** dialog box appears. Click **Finish** to start the remote installation process.
- 9. When the process is finished, click **OK** in the message box that appears. Reboot the remote computer before running the remote instance.

# How to record an unattended installation file (Setup)

The **Record Unattended** Setup option allows you to simulate an installation and create an .iss file that can be used later for an unattended installation of Microsoft® SQL Server<sup>™</sup> 2000. SQL Server files are not installed in this process.

#### To create a file for an unattended installation

- 1. Insert the Microsoft SQL Server 2000 compact disc in your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select **SQL Server 2000 Components**, select **Install Database Server**, and then click **Next** at the Welcome screen of the SQL Server Installation Wizard.
- 3. In the **Computer Name** dialog box, select the option you want, and click **Next**.
- 4. In the **Installation Selection** dialog box, click **Advanced options**, and then in the **Advanced Options** dialog box, click **Record Unattended .ISS file.** Click **Next**.
- 5. In subsequent Setup screens, select the options you want for the unattended installation. After you finish selecting the options, in the **Setup Information** screen, click **Next**.
- 6. In the **Setup Complete** screen, click **Finish**.

This message appears: "Setup has collected the information needed to create an unattended installation file (.iss) for use with later unattended installations of SQL Server."

SQL Server Setup then creates the Setup.iss file in the %windir%

location with the options you have specified.

To run the file, see <u>How to run an unattended installation of SQL</u> <u>Server 2000 (Command Prompt)</u>.

## See Also

Performing an Unattended Installation

# How to run an unattended installation of SQL Server 2000 (Command Prompt)

You can run an unattended installation by using sample batch files and setup initialization files included on the Microsoft® SQL Server<sup>™</sup> 2000 compact disc. Or, you can run the Setup program directly from the command prompt in the appropriate directory for the edition of SQL Server you want to install, using arguments as needed.

### To run an unattended installation using ready-made batch files

- 1. Locate the .bat and .iss files in the root directory of your SQL Server 2000 compact disc.
- 2. View the .bat and associated .iss files, and modify if necessary. For more information, see <u>Creating a Specialized Setup File</u>.
- 3. Run the appropriate batch and setup files from the command prompt:
  - For a standard unattended installation, run Sqlins.bat.
  - For a client-only unattended installation, run Sqlcli.bat.
  - For a custom unattended installation, run Sqlcst.bat.

### To run an unattended installation directly from the command prompt

- 1. Run Setupsql.exe from the Setup directory in the appropriate architecture directory.
- 2. Use arguments as needed:
- -f1 <initialization file path>
   Selects an unattended setup initialization file.

start /wait command (with the -SMS option)

Returns control to the command prompt only after SQL Server Setup completes.

-s flag

Causes the Setup program to run in silent mode with no user interface.

For examples of command prompt options and arguments, see the sample .bat files on your SQL Server 2000 compact disc.

## See Also

Performing an Unattended Installation

How to record an unattended installation file (Setup)

# How to add components to an instance of SQL Server 2000 (Setup)

**Note** You cannot remove components by clearing checkboxes in the **Select Components** dialog box. If you need to remove components from an instance of SQL Server, you must uninstall the instance.

#### To add components to an instance (default or named) of SQL Server 2000

- 1. Run SQL Server Setup, select **SQL Server 2000 Components**, select **Install Database Server**, and then click **Next** at the Welcome screen of the SQL Server Installation Wizard.
- 2. In **Computer Name** dialog box, **Local Computer** is the default option and the local computer name appears in the edit box. Click **Next**.
- 3. In the **Installation Selection** dialog box, click **Upgrade, Remove, or Add Components to an existing instance of SQL Server**, and then click **Next**.
- 4. In the **Instance Name** dialog box, **Default** is selected if you have the Default instance installed. If you want to add components to a named instance, select it from the Instance Name list, and then click **Next**.
- 5. In the **Existing Installation** dialog box, click **Add Components to your existing installation**, and then click **Next**.
- In the Select Components dialog box, select a component from the Components list, and then select items from the related Sub-Components list. Click to select items you want to add, and then click Next.

For information about each component, select the item, and view the **Description** box.

7. When you are finished specifying options, click **Next** in the **Start Copying Files** dialog box to add components to the selected instance of SQL Server.

## See Also

How to uninstall an existing installation of SQL Server (Setup)

# How to access SQL Server Books Online for SQL Server 7.0

If you have Microsoft® SQL Server<sup>™</sup> 7.0 running as the default instance (and SQL Server 2000 as a named instance), SQL Server Books Online for SQL Server 7.0 remains intact on your computer. You can access SQL Server Books Online from the **Start** menu or create a shortcut to it on your desktop.

### To access SQL Server Books Online for SQL Server 7.0 from the Start menu

• On the **Start** menu, point to **Programs** and **Microsoft SQL Server 7.0**, and then click **Books Online**.

#### To create a shortcut to SQL Server Books Online for SQL Server 7.0

- 1. Locate Sqlbol.chm on your computer. (The default location is C:\Mssql7\Books.)
- 2. Right-click Sqlbol.chm, and then click **Create Shortcut**.
- 3. Copy the shortcut to your desktop, where you can use it to access SQL Server Books Online.

To install SQL Server Books Online for SQL Server 7.0 for the first time, or to reinstall it, you must install it from the SQL Server 7.0 compact disc or the SQL Server Web site.

# **To reinstall SQL Server Books Online for SQL Server 7.0 from the SQL Server 7.0 compact disc**

- 1. Insert the Microsoft SQL Server 7.0 compact disc in your CD-ROM drive.
- 2. Locate the file Sqlbol.chm on the compact disc, and copy it to a location on your computer.

3. Create a shortcut on your desktop to SQL Server Books Online for SQL Server 7.0.

# **To download SQL Server Books Online for SQL Server 7.0 from the SQL Server Web site**

- 1. Go to the Microsoft SQL Server Web site, at Microsoft Web site.
- 2. On the SQL Server Welcome page, click **Support**.
- 3. On the Support page, click **Documentation** and follow instructions to access SQL Server 7.0 Books Online.

## How to install English Query (Setup)

#### **To install English Query**

1. Insert the Microsoft® SQL Server<sup>™</sup> 2000 compact disc in your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.

### 2. Select SQL Server 2000 Components.

3. On the Install Components screen, select **Install English Query**.

No further selections are necessary. SQL Server Setup installs English Query on your computer.

## How to install Analysis Services (Setup)

The following procedure is a shortened version of the steps to install Analysis Services. For a more complete installation procedure and for related information, see <u>Running Setup</u>.

#### **To install Analysis Services**

- 1. Insert the Microsoft® SQL Server<sup>™</sup> 2000 compact disc in your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select SQL Server 2000 Components.
- 3. On the Install Components screen, select **Install Analysis Services**.
- 4. At the Welcome screen for Microsoft SQL Server 2000 Analysis Services, click **Next**.
- 5. Follow the directions on the **User Information**, **Software License Agreement**, and related screens.
- 6. In the **Select Components** dialog box, select or clear components as needed, and then click **Next**.

If you want to change the default location of the Analysis Services program files, click **Browse** at **Destination Folder** and select a folder location.

- 7. In the **Data Folder Location** dialog box, accept or change the default location for data files, and then click **Next**.
- 8. In the **Select Program Folder** dialog box, accept or change the default settings, and then click **Next**.

SQL Server Setup installs Analysis Services on your computer.

## How to create a case-sensitive instance of SQL Server 2000 (Setup)

#### To create a case-sensitive instance of SQL Server 2000

- 1. Run SQL Server Setup to install **SQL Server 2000 Components**, select **Install Database Server**, and then click **Next** at the Welcome screen of the SQL Server Installation Wizard.
- 2. In **Computer Name** dialog box, **Local Computer** is the default option and the local computer name appears in the edit box. Click **Next**.
- 3. In the **Installation Selection** dialog box, click click **Create a new instance of SQL Server, or install Client Tools**, and then click **Next**.
- 4. Follow the directions on the **User Information** and related screens.
- 5. In the **Installation Definition** dialog box, click **Server and Client Tools**, and then click **Next**.
- 6. In the **Instance Name** dialog box:
  - To create a case-sensitive default instance, accept the **Default** check box and click **Next**.
  - To create a case-sensitive named instance, clear the **Default** check box and type an instance name.
- 7. In the **Setup Type** dialog box, click **Custom**, and click **Next**.
- 8. In the **Select Components**, **Services Accounts**, and **Authentication Mode** dialog boxes, change or accept the default settings, and then click **Next**.

- 9. In the **Collation Settings** dialog box, you have two options:
  - To make a Windows Locale collation case-sensitive, select **Collation designator** and then select the correct collation designator from the list. Clear the **Binary** check box, and then select the **Case-sensitive** check box.
  - To make a SQL collation case-sensitive, select **SQL Collations**, and then select the correct collation name.

For more information about collation options, click **Help**. When you finish setting the options, click **Next**.

- 10. In subsequent dialog boxes, change or accept the default settings, and then click **Next**.
- 11. When you are finished specifying options, click **Next** in the **Start Copying Files** dialog box.
- 12. In the **Choose Licensing Mode** dialog box, make selections according to your license agreement, and click **Continue** to begin the installation.
- 3. Click **Help** for information about licensing, or see your system administrator.

## See Also

**Collation Settings in Setup** 

## How to set client code pages

To set client code pages under the Windows NT, Windows 98, or Windows 2000 operating systems

• Use the Regional Settings application in Control Panel as described in the Microsoft® Windows NT®, Microsoft Windows® 98, or Microsoft Windows 2000 documentation.

## How to switch from SQL Server 6.5 to SQL Server 2000 (Command Prompt)

#### To switch from SQL Server 6.5 to SQL Server 2000

• Run Vswitch.exe.

### -SwitchTo <65|80>

Determines which version of Microsoft® SQL Server<sup>™</sup> 2000 to activate.

### -Silent <0|1>

Determines if any user interface or messages are displayed. If **1** is specified, a user interface or messages are not displayed. The default is **0**.

## Examples

c:\...\vswitch -SwitchTo 80 -Silent 1

## How to switch from SQL Server 6.5 to SQL Server 2000 (Windows)

#### To switch from SQL Server 6.5 to SQL Server 2000

• On the **Start** menu, point to **Programs/Microsoft SQL Server-Switch**, and then click **Microsoft SQL Server 2000**.

SQL Server Setup switches from Microsoft® SQL Server<sup>™</sup> 2000 version 6.5 to SQL Server 2000.

## How to remove SQL Server 2000 (Windows)

You can remove instances of Microsoft® SQL Server<sup>™</sup> 2000 using Control Panel. Each named instance must be removed separately. When upgrading or maintaining instances, you can remove SQL Server using the Uninstall option in Setup. For more information, see <u>How to uninstall an existing installation</u> (Setup).

You cannot remove a selected component of SQL Server 2000 after it is installed. To remove components, you must remove the entire instance.

#### To remove a named instance of SQL Server 2000

- 1. In **Control Panel**, click **Add/Remove** programs.
- 2. Select a name of an instance of SQL Server 2000, and click **Remove**.

#### To remove all instances of SQL Server 2000

- 1. In **Control Panel**, click **Add/Remove** programs.
- 2. Repeat the removal process for each instance of SQL Server 2000 that is installed.

SQL Server 2000 is uninstalled, but some files may remain. Manually delete directories if any files related to SQL Server 2000 still exist.

## See Also

**Directories and File Locations** 

## **How To Upgrade from SQL Server 6.5**

The How To topics in this section are specific to the process of converting data from Microsoft® SQL Server<sup>™</sup> 6.5 to Microsoft SQL Server 2000 using the SQL Server Upgrade Wizard.

**Note** To run the SQL Server Upgrade Wizard, you must have an instance of Microsoft SQL Server 2000 already installed on your computer.

## **Considerations when upgrading from SQL Server 6.5:**

- During the upgrade process, the SQL Server 6.5 server is stopped and started while objects are scripted and data is extracted. When the data transfer starts, only SQL Server 2000 is running, and it is not possible to access SQL Server 6.5.
- If you are upgrading your existing SQL Server 6.5 server to a different computer that is running SQL Server 2000, both computers should be configured to use a domain user name and password for the MSSQLServer service.
- During this upgrade, user-defined messages created in SQL Server 6.5 using **sp\_addmessage** are not converted to SQL Server 2000. To retain these custom messages, manually copy the messages added in SQL Server 6.5 to your installation of SQL Server 2000.

## See Also

Troubleshooting the SQL Server Upgrade Wizard

Completing the SQL Server Upgrade Wizard

**Upgrade Log Files** 

Upgrading to SQL Server 2000 FAQ

# How to change the size of tempdb in SQL Server 6.5 (ISQL/w)

## To change the size of tempdb in SQL Server 6.5

- 1. On the **Start** menu, point to **Programs/Microsoft SQL Server 6.5**, and then click **ISQL/w**.
- 2. Enter the **sa** password, and then click **Connect**.
- 3. Execute a DISK INIT command to increase the size of the **tempdb** device to at least 25 MB.
- 4. Execute an ALTER DATABASE command to increase the size of the **tempdb** database to at least 25 MB.

## Examples

--Increase the size of the tempdb device DISK INIT name = 'tempdb1',physname = 'c:\mssql\data\tempdb1.DA' GO --Increase the size of tempdb

ALTER DATABASE tempdb ON tempdb1 = 25

## How to change to the current server name in the SQL Server 6.5 master database (ISQL/w)

To change to the current server name in the SQL Server 6.5 master database

- Start Microsoft<sup>®</sup> SQL Server<sup>™</sup> in minimal configuration mode. In a command prompt window, from the \Mssql\Binn directory, run: sqlservr -f
- 2. On the **Start** menu, point to **Programs** /**Microsoft SQL Server 6.5**, and then click **ISQL**/w.
- 3. Enter the **sa** password, and then click **Connect**.
- 4. Execute SELECT @@SERVERNAME to retrieve the former server name.
- 5. Execute **sp\_dropserver** to drop the former server.
- 6. Execute **sp\_addserver** to add the current server.
- 7. Stop SQL Server. In the command prompt window, press Ctrl+C.
- 8. Restart SQL Server.
- 9. Execute SELECT @@SERVERNAME to verify the current server name.

## Examples

--Start SQL Server in minimal configuration mode.

--Retrieve the former server name.

SELECT @@SERVERNAME

--Drop the server returned from the previous select.

sp\_dropserver 'SERVER6X'

--Add the current server.

sp\_addserver 'SERVER70', local

--Stop SQL Server.

--Restart SQL Server in minimal configuration mode.

--Verify the current server name.

SELECT @@SERVERNAME

## How to update the device file locations in the SQL Server 6.5 master database (ISQL/w)

To update the device file locations in the SQL Server 6.5 master database

- 1. On the **Start** menu, point to **Programs/Microsoft SQL Server 6.5**, and then click **ISQL/w**.
- 2. Enter the **sa** password, and then click Connect.
- 3. Select from **sysdevices** in the **master** database to view the old device file locations.
- 4. Execute **sp\_configure** to allow updates to the system tables, and then reconfigure with override.
- 5. Update the device file locations that have changed.
- 6. Execute **sp\_configure** to disallow updates to the system tables, and then reconfigure with override.

## Examples

--View the old device file locations SELECT phyname FROM sysdevices

--Allow updates to the system tables sp\_configure 'allow updates',1 GO RECONFIGURE WITH OVERRIDE GO

```
---Update device file locations that have changed
UPDATE sysdevices
SET phyname = "E:\Data\HR\HR1.dat"
WHERE name = "HumanResources1"
GO
UPDATE sysdevices
SET phyname = "E:\Data\HR\HR1Log.dat"
WHERE name = "HumanResources1Log"
GO
--Disallow updates to the system tables
sp_configure 'allow updates',0
GO
RECONFIGURE WITH OVERRIDE
GO
```

## How to estimate the disk space required for an upgrade from SQL Server version 6.5 to SQL Server 2000 (SQL Server Upgrade Wizard)

**Note** To run the SQL Server Upgrade Wizard, you must have an instance of Microsoft<sup>®</sup> SQL Server<sup>™</sup> 2000 already installed on your computer.

#### To estimate the disk space required for an upgrade

- 1. On the **Start** menu, point to **Programs/Microsoft SQL Server-Switch**, click **SQL Server Upgrade Wizard**, and then click **Next**.
- 2. Select **Named pipe**; then click **Next**.
- 3. In **Export server (6.5)**, in the **Server name** box, enter the name of the local or remote computer on which SQL Server 6.5 resides.
- 4. In the **Administrator password ('sa')** box, enter the **sa** password for SQL Server 6.5, and then click **Next**.

Unless you have changed it, the system administrator password for SQL Server 2000 is blank.

- 5. Include the databases to upgrade. Move any database not to include in the disk space estimation to the Exclude list, and then click **Next**.
- 6. Select **Use the default configuration or edit the default**; then click **Edit**.

The SQL Server Upgrade Wizard layout utility appears, showing the proposed layout of the SQL Server 2000 data files.

7. Click **Advanced**.

- 8. Click an object in the **Proposed database layout** box to view details in the **Object details** box.
- 9. The **Drive summary** box shows the estimated size of all SQL Server 2000 data files and the free disk space left on all of the local fixed disks. On the **Options** menu, select **Freespace includes 6.5 files** to view the free space that would exist if the SQL Server 6.5 data files were deleted.
- 10. Click **Accept** to return to the **Database Creation** dialog box.
- 11. Click **Cancel** to quit the SQL Server Upgrade Wizard.

## How to edit the default database configuration (SQL Server Upgrade Wizard)

**Note** To run the SQL Server Upgrade Wizard, you must have an instance of Microsoft® SQL Server<sup>™</sup> 2000 already installed on your computer.

### To edit the default database configuration

- 1. In the **Database Creation** dialog box of the SQL Server Upgrade Wizard, click **Edit**.
- 2. Click **Advanced** to view object details and drive summaries.
- 3. In the **Proposed database layout** box, double-click a database file.
- 4. Change any database file attributes, and then click **OK**.
- 5. View the changes to the drive summary.
- 6. When all changes have been made, click **Accept** to save the database configuration.

## See Also

Proposed Database Layout

## How to perform a SQL Server version 6.5 to SQL Server 2000 upgrade using a direct pipeline (SQL Server Upgrade Wizard)

**Note** To run the SQL Server Upgrade Wizard, you must have an instance of Microsoft® SQL Server<sup>™</sup> 2000 already installed on your computer.

## To perform a SQL Server version 6.5 to SQL Server 2000 upgrade by named pipe

- 1. On the **Start** menu, point to **Programs/Microsoft SQL Server-Switch**, click **SQL Server Upgrade Wizard**, and then click **Next**.
- 2. In the Data and Object Transfer screen, accept the default selections, including **Named pipe**, and then click **Next**. Verification options are recommended, but not required. Click **Help** for information.
- On the Logon screen, in the Server name box in the Export server (6.5) group box, enter the name of the local or remote computer on which Microsoft SQL Server version 6.5 is installed.

In the **Administrator password ('sa')** box, enter the **sa** password for SQL Server 6.5, and then click **Next**. Unless you have changed it, the system administrator password for SQL Server 2000 is blank.

For **Import Server (2000)**, the server name is filled in. Enter the optional startup arguments, if you want. Click **Help** for information. When you are finished setting options, click **Next**.

- 4. In the message box asking if you want to continue, click **Yes** if you are ready to upgrade. The SQL Server Upgrade Wizard shuts down SQL Server 6.5 and starts SQL Server 2000.
- 5. In the **Code Page Selection** screen, accept or change the default settings, and then click **Next**.

6. In the **Database Selection** screen, include the databases to upgrade. Move any databases you do not want upgraded at this time to the **Exclude** list, and then click **Next**.

Converting all databases is recommended.

7. In the **Database Creation** dialog box, select **Use the default configuration or edit the default**, and then click **Next**.

Click **Edit** to examine and make changes to the proposed disk configuration within the layout utility. In the **Proposed Database Layout** box, make changes as needed. Click **Advanced** to view **Object Details** and **Drive Summary**. When you are finished, click **Accept** to return to the SQL Server Upgrade Wizard.

- 8. In the **System Configuration** screen, in **System objects to transfer**, select the object types to transfer from SQL Server 6.5 to SQL Server 2000:
  - Server configuration

Login and remote login registrations and server configuration options relevant to SQL Server 2000 are transferred as part of the version upgrade.

#### • Replication settings

All articles, subscriptions and publications of each selected database, plus the distribution database, if any, are transferred and upgraded.

#### • SQL Executive settings

All tasks scheduled by SQL Executive are transferred and upgraded so that SQL Server 2000 can schedule and run those tasks in SQL Server Agent.

- 9. In the System Configuration screen, in Advanced settings, for ANSI Nulls, select:
  - Off, if ANSI nulls should not be used when stored procedures

are created. This is the default.

- **On**, if ANSI nulls should be used when stored procedures are created.
- 10. In **Quoted identifiers**, select one of these options, and then click **Next**:
  - **Mixed (or don't know)**, if some of your objects were created with QUOTED\_IDENTIFIER set to ON and others with it set to OFF, or if you are not sure how they were created.
  - **Off**, if all objects should be compiled with QUOTED\_IDENTIFIER set to OFF.
  - **On**, if all objects should be compiled with QUOTED\_IDENTIFIER set to ON.
- 11. In the **Completing the SQL Server Wizard** screen, view the summary of choices you have made. Click **View warnings and choices in notepad** to open a text version of the upgrade script. If all options are correct, click **Finish**.

The **SQL Server Upgrade Script Interpreter** screen appears, with information on the progress of the upgrade.

## See Also

Order of Upgrade Using a Direct Pipeline or Tape Drive

## How to perform a SQL Server version 6.5 to SQL Server 2000 upgrade using a tape drive (SQL Server Upgrade Wizard)

**Note** To run the SQL Server Upgrade Wizard, you must have an instance of Microsoft® SQL Server<sup>™</sup> 2000 already installed on your computer.

To perform a SQL Server version 6.5 to SQL Server 2000 upgrade using a tape drive

- 1. On the **Start** menu, point to **Programs/Microsoft SQL Server-Switch**, click **SQL Server Upgrade Wizard**, and then click **Next**.
- 2. In the **Data and Object Transfer** screen, click **Tape**, and then click **Next**. Verification options are recommended, but not required. Click **Help** for information.
- On the Logon screen, in the Server name box in the Export server (6.5) group box, enter the name of the computer on which Microsoft SQL Server version 6.5 is installed.

In the **Administrator password ('sa')** box, enter the **sa** password for SQL Server 6.5, and then click **Next**.

Unless you have changed it, the **sa** password for SQL Server 2000 is blank.

For **Import Server (2000)**, the server name is filled in. Enter optional startup arguments, if you want. Click **Help** for information. When you are finished setting options, click **Next**.

4. In the message box asking if you want to continue, click **Yes** if you are ready to upgrade. The SQL Server Upgrade Wizard switches to the SQL Server 2000 server.

- 5. In the Code Page Selection screen, accept or change the default settings, and then click **Next**.
- 6. In the Database Selection screen, include the databases to upgrade. Move any database not to be upgraded at this time to the Exclude list, and then click **Next**.
- 7. In **Device for data transfer**, specify the location of the tape drive.
- 8. In **6.5 device backup options**, select **Backup 6.5 devices before exporting data** if you have not backed up the databases already.

Prior to creating the SQL Server 2000 databases, the SQL Server Upgrade Wizard either prompts you to back up the SQL Server 6.5 devices or copies the devices for you automatically.

9. Select **Delete 6.5 devices before importing data** if necessary due to lack of disk space, and then click **Next**.

After objects and data are exported, and before creating databases in SQL Server 2000, the SQL Server Upgrade Wizard deletes the SQL Server 6.5 devices to reclaim disk space.

10. Select **Use the default configuration** or edit the default, and then click **Next**.

Click **Edit** to examine and make changes to the proposed disk configuration within the layout utility. In the **Proposed Database Layout** box, make changes as needed. Click **Advanced** to view **Object Details** and **Drive Summary**. When you are finished, click **Accept** to return to the SQL Server Upgrade Wizard.

- 11. In **System objects to transfer**, select the object types to transfer from SQL Server 6.5 to SQL Server 2000:
  - Server configuration

Login and remote login registrations and server configuration options relevant to SQL Server 2000 are transferred as part of

the version upgrade.

## • Replication settings

All articles, subscriptions, and publications of each selected database, plus the distribution database, if any, are transferred and upgraded.

### • SQL Executive settings

All tasks scheduled by SQL Executive are transferred and upgraded so that SQL Server 2000 can schedule and run those tasks in SQL Server Agent.

- 12. In ANSI Nulls, select:
  - **Off**, if ANSI nulls should not be used when stored procedures are created. This is the default.
  - **On**, if ANSI nulls should be used when stored procedures are created.
- 13. In **Quoted Identifiers**, select one of these options, and then click **Next**:
  - **Mixed (or don't know)**, if some of your objects were created with QUOTED\_IDENTIFIER set to ON and others with it set to OFF, or if you are not sure how they were created.
  - **Off**, if all objects should be compiled with QUOTED\_IDENTIFIER set to OFF.
  - **On**, if all objects should be compiled with QUOTED\_IDENTIFIER set to ON.
- 14. In the **Completing the SQL Server Wizard** screen, view the summary of choices you have made. Click **View warnings and choices in notepad** to open a text version of the upgrade script. If all options are correct, click **Finish**.

The **SQL Server Upgrade Script Interpreter** screen appears with information about the progress of the upgrade.

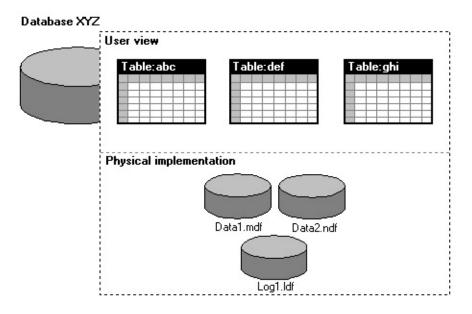
## See Also

Order of Upgrade Using a Direct Pipeline or Tape Drive

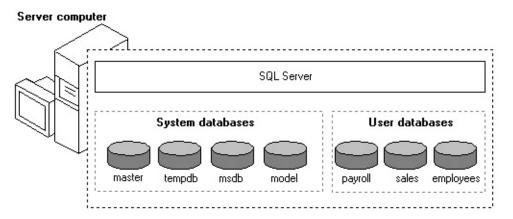
## **Database Architecture**

Microsoft® SQL Server<sup>™</sup> 2000 data is stored in databases. The data in a database is organized into the logical components visible to users. A database is also physically implemented as two or more files on disk.

When using a database, you work primarily with the logical components such as tables, views, procedures, and users. The physical implementation of files is largely transparent. Typically, only the database administrator needs to work with the physical implementation.



Each instance of SQL Server has four system databases (**master**, **model**, **tempdb**, and **msdb**) and one or more user databases. Some organizations have only one user database, containing all the data for their organization. Some organizations have different databases for each group in their organization, and sometimes a database used by a single application. For example, an organization could have one database for sales, one for payroll, one for a document management application, and so on. Sometimes an application uses only one database; other applications may access several databases.



It is not necessary to run multiple copies of the SQL Server database engine to allow multiple users to access the databases on a server. An instance of the SQL Server Standard or Enterprise Edition is capable of handling thousands of users working in multiple databases at the same time. Each instance of SQL Server makes all databases in the instance available to all users that connect to the instance, subject to the defined security permissions.

When connecting to an instance of SQL Server, your connection is associated with a particular database on the server. This database is called the current database. You are usually connected to a database defined as your default database by the system administrator, although you can use connection options in the database APIs to specify another database. You can switch from one database to another using either the Transact-SQL USE *database\_name* statement, or an API function that changes your current database context.

SQL Server 2000 allows you to detach databases from an instance of SQL Server, then reattach them to another instance, or even attach the database back to the same instance. If you have a SQL Server database file, you can tell SQL Server when you connect to attach that database file with a specific database name.

## See Also

Database Design Considerations

## **Logical Database Components**

The data in a Microsoft® SQL Server<sup>™</sup> 2000 database is organized into several different objects. These objects are what a user can see when they connect to the database.

In SQL Server 2000, these components are defined as objects:

Constraints	Tables
Defaults	Triggers
Indexes	User-defined data types
Keys	User-defined functions
Stored procedures	Views

## **Data Types and Table Structures**

All the data in Microsoft® SQL Server<sup>™</sup> 2000 databases is contained in objects called tables. Each table represents some type of object meaningful to the users. For example, in a school database you would find tables such as a class table, an instructor table, and a student table.

SQL Server tables have two main components:

• Columns

Each column represents some attribute of the object modeled by the table, such as a parts table having columns for ID, color, and weight.

• Rows

Each row represents an individual occurrence of the object modeled by the table. For example, the parts table would have one row for each part carried by the company.

parts table		
ID	color	weight
AB123	Blue	10.5
CD 456	Red	8.0
EF789	Green	9.25
GH012	Yellow	8.0
IJ341	Blue	1.0

### **Data Types**

Because each column represents one attribute of an object, the data in each occurrence of the column is similar. One of the properties of a column is called its data type, which defines the type of data the column can hold. SQL Server has several base data types that can be specified for columns:

binary	Bigint	bit	Char	datetime
decimal	Float	image	Int	Money

nchar	Ntext	nvarchar	Numeric	Real
smalldatetime	smallint	smallmoney	sql_variant	sysname
text	timestamp	tinyint	varbinary	varchar
uniqueidentifier				

SQL Server 2000 also supports a **table** base data type, which can be used to store the result set of an SQL statement. The **table** data type cannot be used for columns in a table. It can only be used for Transact-SQL variables and the return values of user-defined functions. For more information, see <u>Using Special Data</u>.

Users can also create their own user-defined data types, for example:

-- Create a birthday data type that allows nulls. EXEC sp\_addtype birthday, datetime, 'NULL' GO -- Create a table using the new data type. CREATE TABLE employee (emp\_id char(5), emp\_first\_name char(30), emp\_last\_name char(40), emp\_birthday birthday)

A user-defined data type makes a table structure more meaningful to programmers and helps ensure that columns holding similar classes of data have the same base data type.

SQL Server provides several data type synonyms to help support SQL-92 data type names not included as base data types, such as national character and character varying. When a synonym is specified in a CREATE TABLE statement, the column is assigned the base data type associated with the synonym. For more information, see Data Type Synonyms.

A <u>domain</u> is the set of all allowable values in a column. It includes not only the concept of enforcing data types, but also the values allowed in the column. For example, a part color domain would include both the data type, such as **char(6)**, and the character strings allowed in the column, such as Red, Blue, Green, Yellow, Brown, Black, White, Teal, Grey, and Silver. Domain values can be enforced through mechanisms such as CHECK constraints and triggers.

When a column has been assigned a data type, all values placed into the column must be of that data type. SQL statements can specify that values of different data types be used as the source value only if SQL Server can implicitly convert the source value data type to the data type of the column. For example, SQL Server supports the implicit conversion of **int** values to **decimal**; therefore, SQL statements can specify **int** values as the value to be assigned to a **decimal** column.

The SQL Server 2000 **sql\_variant** data type is a special data type that allows you to store values of multiple base data types in the same column. For example, you can store **nchar** values, **int** values, and **decimal** values in the same column. For more information, see <u>Using sql\_variant Data</u>.

### **Null Values**

Columns can either accept or reject null values. NULL is a special value in databases that represents the concept of an unknown value. NULL is not the same as a blank character or 0. Blank is actually a valid character, and 0 is a valid number. NULL simply represents the idea that we do not know what this value is. NULL is also different from a zero-length string. If a column definition contains the NOT NULL clause, you cannot insert rows having the value NULL for that row. If the column definition has only the NULL keyword, it accepts NULL values.

Allowing NULL values in a column can increase the complexity of any logical comparisons using the column. The SQL-92 standard states that any comparison against a NULL value does not evaluate to TRUE or FALSE, it evaluates to UNKNOWN. This introduces three-value logic to comparison operators, which can be difficult to manage correctly.

### **System Tables**

SQL Server stores the data defining the configuration of the server and all its tables in a special set of tables known as system tables. Users should not query or update the system tables directly unless there is no other way to get the data required by the application. Only SQL Server should reference the system tables in response to administration commands issued by users. The system tables can change from version to version; applications referencing system tables directly may have to be rewritten before they can be upgraded to a newer version of SQL

Server with a different version of the system tables. SQL Server exposes most of the information from the system tables through other means. For more information, see <u>System Tables</u>.

### **Temporary Tables**

SQL Server supports temporary tables. These tables have names that start with a number sign (#). If a temporary table is not dropped when a user disconnects, SQL Server automatically drops the temporary table. Temporary tables are not stored in the current database; they are stored in the **tempdb** system database.

There are two types of temporary tables:

• Local temporary tables

The names of these tables begin with one number sign (#). These tables are visible only to the connection that created them.

• Global temporary tables

The names of these tables begin with two number signs (##). These tables are visible to all connections. If the tables are not dropped explicitly before the connection that created them disconnects, they are dropped as soon as all other tasks stop referencing them. No new tasks can reference a global temporary table after the connection that created it disconnects. The association between a task and a table is always dropped when the current statement completes executing; therefore, global temporary tables are usually dropped soon after the connection that created them disconnects.

Many traditional uses of temporary tables can now be replaced with variables that have the **table** data type.

#### **Working with Tables**

Users work with the data in tables using data manipulation language (DML) SQL statements:

-- Get a list of all employees named Smith: SELECT emp\_first\_name, emp\_last\_name

FROM employee WHERE emp\_last\_name = 'Smith'

-- Delete an employee who quit: DELETE employee WHERE emp\_id = 'OP123'

-- Add a new employee: INSERT INTO employee VALUES ( 'OP456', 'Dean', 'Straight', '01/01/1960')

-- Change an employee name: UPDATE employee SET emp\_last\_name = 'Smith' WHERE emp\_id = 'OP456'

#### See Also

Specifying a Column Data Type

<u>Tables</u>

## **SQL Views**

A view can be thought of as either a virtual table or a stored query. The data accessible through a view is not stored in the database as a distinct object. What is stored in the database is a SELECT statement. The result set of the SELECT statement forms the virtual table returned by the view. A user can use this virtual table by referencing the view name in Transact-SQL statements the same way a table is referenced. A view is used to do any or all of these functions:

• Restrict a user to specific rows in a table.

For example, allow an employee to see only the rows recording his or her work in a labor-tracking table.

• Restrict a user to specific columns.

For example, allow employees who do not work in payroll to see the name, office, work phone, and department columns in an employee table, but do not allow them to see any columns with salary information or personal information.

- Join columns from multiple tables so that they look like a single table.
- Aggregate information instead of supplying details.

For example, present the sum of a column, or the maximum or minimum value from a column.

Views are created by defining the SELECT statement that retrieves the data to be presented by the view. The data tables referenced by the SELECT statement are known as the base tables for the view. In this example, **titleview** in the **pubs** database is a view that selects data from three base tables to present a virtual table of commonly needed data:

CREATE VIEW titleview AS SELECT title, au\_ord, au\_lname, price, ytd\_sales, pub\_id FROM authors AS a JOIN titleauthor AS ta ON (a.au\_id = ta.au\_id) JOIN titles AS t ON (t.title\_id = ta.title\_id)

You can then reference **titleview** in statements in the same way you would reference a table:

#### SELECT \* FROM titleview

A view can reference another view. For example, **titleview** presents information that is useful for managers, but a company typically discloses year-to-date figures only in quarterly or annual financial statements. A view can be built that selects all the **titleview** columns except **au\_ord** and **ytd\_sales**. This new view can be used by customers to get lists of available books without seeing the financial information:

CREATE VIEW Cust\_titleview AS SELECT title, au\_lname, price, pub\_id FROM titleview

Views can be used to partition data across multiple databases or instances of Microsoft® SQL Server<sup>™</sup> 2000. Partitioned views can be used to distribute database processing across a group of servers. The group of servers has the same performance benefits as a cluster of servers, and can be used to support the processing needs of the largest Web sites or corporate data centers. An original table is subdivided into several member tables, each of which has a subset of the rows from the original table. Each member table can be placed in databases on separate servers. Each server also gets a partitioned view. The partitioned view uses the Transact-SQL UNION operator to combine the results of selects against all the member tables into a single result set that behaves exactly like a copy of the full original table. For example, a table is partitioned across three servers. On the first server you define a partitioned view similar to this:

## CREATE VIEW PartitionedView AS SELECT \*

FROM MyDatabase.dbo.PartitionTable1

#### UNION ALL SELECT \* FROM Server2.MyDatabase.dbo.PartitionTable2 UNION ALL SELECT \* FROM Server3.MyDatabase.dbo.PartitionTable3

You define similar partitioned views on each of the other servers. With these three views, any Transact-SQL statements on any of the three servers that reference **PartitionedView** will see the same behavior as from the original table. It is as if a copy of the original table exists on each server, when in fact there is only one member table and a partitioned view on each table. For more information, see <u>Scenarios for Using Views</u>.

Views in all versions of SQL Server are updatable (can be the target of UPDATE, DELETE, or INSERT statements), as long as the modification affects only one of the base tables referenced by the view, for example:

```
-- Increase the prices for publisher '0736' by 10%.
UPDATE titleview
SET price = price * 1.10
WHERE pub_id = '0736'
GO
```

SQL Server 2000 supports more complex types of INSERT, UPDATE, and DELETE statements that reference views. INSTEAD OF triggers can be defined on a view to specify the individual updates that must be performed against the base tables to support the INSERT, UPDATE, or DELETE statement. Also, partitioned views support INSERT, UDPATE, and DELETE statements that modify multiple member tables referenced by the view.

Indexed views are a SQL Server 2000 feature that greatly improves the performance of complex views of the type usually found in data warehouses or other decision support systems.

Views are called virtual tables because the result set of a view is us not usually saved in the database The result set for a view is dynamically incorporated into the logic of the statement and the result set is built dynamically at run time. For more information, see <u>View Resolution</u>.

Complex queries, such as those in decision support systems, can reference large numbers of rows in base tables, and aggregate large amounts of information into relatively concise aggregates such as sums or averages. SQL Server 2000 supports creating a clustered index on a view that implements such a complex query. When the CREATE INDEX statement is executed the result set of the view SELECT is stored permanently in the database. Future SQL statements that reference the view will have substantially better response times. Modifications to the base data are automatically reflected in the view.

The SQL Server 2000 CREATE VIEW statement supports a SCHEMABINDING option that prevents the tables referenced by the view being changed without adjusting the view. You must specify SCHEMABINDING for any view on which you create an index.

#### See Also

<u>CREATE INDEX</u> <u>CREATE TRIGGER</u> <u>CREATE VIEW</u> <u>Designing an Indexed View</u> Views

## **SQL Stored Procedures**

A <u>stored procedure</u> is a group of Transact-SQL statements compiled into a single execution plan.

Microsoft® SQL Server<sup>™</sup> 2000 stored procedures return data in four ways:

- Output parameters, which can return either data (such as an integer or character value) or a cursor variable (cursors are result sets that can be retrieved one row at a time).
- Return codes, which are always an integer value.
- A result set for each SELECT statement contained in the stored procedure or any other stored procedures called by the stored procedure.
- A global cursor that can be referenced outside the stored procedure.

Stored procedures assist in achieving a consistent implementation of logic across applications. The SQL statements and logic needed to perform a commonly performed task can be designed, coded, and tested once in a stored procedure. Each application needing to perform that task can then simply execute the stored procedure. Coding business logic into a single stored procedure also offers a single point of control for ensuring that business rules are correctly enforced.

Stored procedures can also improve performance. Many tasks are implemented as a series of SQL statements. Conditional logic applied to the results of the first SQL statements determines which subsequent SQL statements are executed. If these SQL statements and conditional logic are written into a stored procedure, they become part of a single execution plan on the server. The results do not have to be returned to the client to have the conditional logic applied; all of the work is done on the server. The IF statement in this example shows embedding conditional logic in a procedure to keep from sending a result set to the application:

#### IF (@QuantityOrdered < (SELECT QuantityOnHand

```
FROM Inventory
WHERE PartID = @PartOrdered))
```

BEGIN

-- SQL statements to update tables and process order.

END

ELSE

BEGIN

-- SELECT statement to retrieve the IDs of alternate items

```
-- to suggest as replacements to the customer.
```

END

Applications do not need to transmit all of the SQL statements in the procedure: they have to transmit only an EXECUTE or CALL statement containing the name of the procedure and the values of the parameters.

Stored procedures can also shield users from needing to know the details of the tables in the database. If a set of stored procedures supports all of the business functions users need to perform, users never need to access the tables directly; they can just execute the stored procedures that model the business processes with which they are familiar.

An illustration of this use of stored procedures is the SQL Server system stored procedures used to insulate users from the system tables. SQL Server includes a set of system stored procedures whose names usually start with **sp\_**. These system stored procedures support all of the administrative tasks required to run a SQL Server system. You can administer a SQL Server system using the Transact-SQL administration-related statements (such as CREATE TABLE) or the system stored procedures, and never need to directly update the system tables.

## **Stored Procedures and Execution Plans**

In SQL Server version 6.5 and earlier, stored procedures were a way to partially precompile an execution plan. At the time the stored procedure was created, a partially compiled execution plan was stored in a system table. Executing a stored procedure was more efficient than executing an SQL statement because SQL Server did not have to compile an execution plan completely, it only had to

finish optimizing the stored plan for the procedure. Also, the fully compiled execution plan for the stored procedure was retained in the SQL Server procedure cache, meaning that subsequent executions of the stored procedure could use the precompiled execution plan.

SQL Server 2000 and SQL Server version 7.0 incorporate a number of changes to statement processing that extend many of the performance benefits of stored procedures to all SQL statements. SQL Server 2000 and SQL Server 7.0 do not save a partially compiled plan for stored procedures when they are created. A stored procedure is compiled at execution time, like any other Transact-SQL statement. SQL Server 2000 and SQL Server 7.0 retain execution plans for all SQL statements in the procedure cache, not just stored procedure execution plans. The database engine uses an efficient algorithm for comparing new Transact-SQL statements with the Transact-SQL statements of existing execution plans. If the database engine determines that a new Transact-SQL statement matches the Transact-SQL statement of an existing execution plan, it reuses the plan. This reduces the relative performance benefit of precompiling stored procedures by extending execution plan reuse to all SQL statements.

SQL Server 2000 and SQL Server version 7.0 offer new alternatives for processing SQL statements. For more information, see <u>Query Processor</u> <u>Architecture</u>.

### **Temporary Stored Procedures**

SQL Server 2000 also supports temporary stored procedures that, like temporary tables, are dropped automatically when you disconnect. Temporary stored procedures are stored in **tempdb** and are useful when connected to earlier versions of SQL Server. Temporary stored procedures can be used when an application builds dynamic Transact-SQL statements that are executed several times. Rather than have the Transact-SQL statements recompiled each time, you can create a temporary stored procedure that is compiled on the first execution, and then execute the precompiled plan multiple times. Heavy use of temporary stored procedures, however, can lead to contention on the system tables in **tempdb**.

Two features of SQL Server 2000 and SQL Server 7.0 eliminate the need for using temporary stored procedures:

- Execution plans from prior SQL statements can be reused. This is especially powerful when coupled with the use of the new **sp\_executesql** system stored procedure.
- Natively support for the prepare/execute model of OLE DB and ODBC without using any stored procedures.

For more information about alternatives to using temporary stored procedures, see <u>Execution Plan Caching and Reuse</u>.

#### **Stored Procedure Example**

This simple stored procedure example illustrates three ways stored procedures can return data:

- 1. It first issues a SELECT statement that returns a result set summarizing the order activity for the stores in the **sales** table.
- 2. It then issues a SELECT statement that fills an output parameter.
- 3. Finally, it has a RETURN statement with a SELECT statement that returns an integer. Return codes are generally used to pass back error checking information. This procedure runs without errors, so it returns another value to illustrate how returned codes are filled.

USE Northwind

GO

DROP PROCEDURE OrderSummary

GO

CREATE PROCEDURE OrderSummary @MaxQuantity INT OUTPU -- SELECT to return a result set summarizing

-- employee sales.

SELECT Ord.EmployeeID, SummSales = SUM(OrDet.UnitPrice \* Or FROM Orders AS Ord

JOIN [Order Details] AS OrDet ON (Ord.OrderID = OrDet.OrderII

### GROUP BY Ord.EmployeeID ORDER BY Ord.EmployeeID

-- SELECT to fill the output parameter with the -- maximum quantity from Order Details. SELECT @MaxQuantity = MAX(Quantity) FROM [Order Details]

-- Return the number of all items ordered. RETURN (SELECT SUM(Quantity) FROM [Order Details]) GO

-- Test the stored procedure.

-- DECLARE variables to hold the return code -- and output parameter. DECLARE @OrderSum INT DECLARE @LargestOrder INT

-- Execute the procedure, which returns

-- the result set from the first SELECT.

EXEC @OrderSum = OrderSummary @MaxQuantity = @LargestOrd

-- Use the return code and output parameter. PRINT 'The size of the largest single order was: ' + CONVERT(CHAR(6), @LargestOrder) PRINT 'The sum of the quantities ordered was: ' + CONVERT(CHAR(6), @OrderSum) GO

The output from running this sample is:

EmployeeID SummSales

-----

1 202,143.71

2	177,749.26
	,

- 3 213,051.30
- 4 250,187.45
- 5 75,567.75
- 6 78,198.10
- 7 141,295.99
- 8 133,301.03
- 9 82,964.00

The size of the largest single order was: 130 The sum of the quantities ordered was: 51317

## See Also

Stored Procedures

## **SQL User-Defined Functions**

Functions in programming languages are subroutines used to encapsulate frequently performed logic. Any code that must perform the logic incorporated in a function can call the function rather than having to repeat all of the function logic.

Microsoft<sup>®</sup> SQL Server<sup>™</sup> 2000 supports two types of functions:

• Built-in functions

Operate as defined in the Transact-SQL Reference and cannot be modified. The functions can be referenced only in Transact-SQL statements using the syntax defined in the Transact-SQL Reference. For more information about these built-in functions, see <u>Using Functions</u>.

• User-defined functions

Allow you to define your own Transact-SQL functions using the CREATE FUNCTION statement. For more information about these built-in functions, see <u>User-defined Functions</u>.

User-defined functions take zero or more input parameters, and return a single value. Some user-defined functions return a single, scalar data value, such as an **int**, **char**, or **decimal** value.

For example, this statement creates a simple function that returns a decimal:

**CREATE FUNCTION CubicVolume** 

-- Input dimensions in centimeters.

(@CubeLength decimal(4,1), @CubeWidth decimal(4,1),

@CubeHeight decimal(4,1) )

RETURNS decimal(12,3) -- Cubic Centimeters.

AS

BEGIN

RETURN ( @CubeLength \* @CubeWidth \* @CubeHeight ) END This function can then be used anywhere an integer expression is allowed, such as in a computed column for a table:

```
CREATE TABLE Bricks
 (
  BrickPartNmbr int PRIMARY KEY,
  BrickColor
               nchar(20),
  BrickHeight
                decimal(4,1),
                decimal(4,1),
  BrickLength
  BrickWidth
                decimal(4,1),
  BrickVolume AS
        (
        dbo.CubicVolume(BrickHeight,
              BrickLength, BrickWidth)
        )
 )
```

SQL Server 2000 also supports user-defined functions that return a **table** data type:

- A function can declare an internal **table** variable, insert rows into the variable, and then return the variable as its return value.
- A class of user-defined functions known as in-line functions, return the result set of a SELECT statement as a variable of type **table**.

These functions can be used in places where table expressions can be specified. For more information about the **table** data type, see <u>Using Special Data</u>.

User-defined functions that return a **table** can be powerful alternatives to views. A user-defined function that returns a **table** can be used where table or view expressions are allowed in Transact-SQL queries. Views are limited to a single SELECT statement; however, user-defined functions can contain additional statements that allow more powerful logic than is possible in views.

A user-defined function that returns a **table** can also replace stored procedures that return a single result set. The **table** returned by a user-defined function can

be referenced in the FROM clause of a Transact-SQL statement, whereas stored procedures that return result sets cannot. For example, **fn\_EmployeesInDept** is a user-defined function that returns a **table** and can be invoked by a SELECT statement:

```
SELECT *
FROM tb_Employees AS E,
dbo.fn_EmployeesInDept('shipping') AS EID
WHERE E.EmployeeID = EID.EmployeeID
```

This is an example of a statement that creates a function in the **Northwind** database that will return a **table**:

```
CREATE FUNCTION LargeOrderShippers ( @FreightParm money )
RETURNS @OrderShipperTab TABLE
```

```
(
  ShipperID
             int,
  ShipperName nvarchar(80),
  OrderID
             int,
  ShippedDate datetime,
  Freight
            money
 )
AS
BEGIN
 INSERT @OrderShipperTab
    SELECT S.ShipperID, S.CompanyName,
        O.OrderID, O.ShippedDate, O.Freight
    FROM Shippers AS S
       INNER JOIN Orders AS O ON (S.ShipperID = O.ShipVia)
    WHERE O.Freight > @FreightParm
 RETURN
END
```

In this function, the local return variable name is **@OrderShipperTab**. Statements in the function build the **table** result returned by the function by inserting rows into the variable **@OrderShipperTab**. External statements invoke the function to reference the **table** returned by the function:

SELECT \*
FROM LargeOrderShippers( \$500 )

## **Constraints, Rules, Defaults, and Triggers**

Table columns have properties other than data type and size. These other properties are an important part in ensuring the integrity of data in a database:

• Data integrity refers to each occurrence of a column having a correct data value.

The data values must be of the right data type and in the correct domain.

• Referential integrity indicates that the relationships between tables have been properly maintained.

Data in one table should only point to existing rows in another table; it should not point to rows that do not exist.

Objects used to maintain both types of integrity include:

- Constraints
- Rules
- Defaults
- Triggers

## Constraints

Constraints allow you to define the way Microsoft® SQL Server<sup>™</sup> 2000 automatically enforces the integrity of a database. Constraints define rules regarding the values allowed in columns and are the standard mechanism for enforcing integrity. Using constraints is preferred to using triggers, rules, and defaults. The query optimizer also uses constraint definitions to build high-performance query execution plans.

### **Classes of Constraints**

SQL Server 2000 supports five classes of constraints.

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

A CHECK constraint specifies a Boolean (evaluates to TRUE or FALSE) search condition that is applied to all values entered for the column; all values that do not evaluate to TRUE are rejected. You can specify multiple CHECK constraints for each column. This sample shows the creation of a named constraint, **chk\_id**, that further enforces the domain of the primary key by ensuring that only numbers within a specified range are entered for the key.

CREATE TABLE cust\_sample

```
(

cust_id int PRIMARY KEY,

cust_name char(50),

cust_address char(50),

cust_credit_limit money,

CONSTRAINT chk_id CHECK (cust_id BETWEEN 0 and 1)

)
```

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

No two rows in the table are allowed to have the same not null values for the columns in a UNIQUE constraint. Primary keys also enforce uniqueness, but primary keys do not allow null values. A UNIQUE constraint is preferred over a unique index.

• PRIMARY KEY constraints identify the column or set of columns whose values uniquely identify a row in a table.

No two rows in a table can have the same primary key value. You cannot enter a NULL for any column in a primary key. NULL is a special value in databases that represents an unknown value, which is distinct from a blank or 0 value. Using a small, integer column as a primary key is recommended. Each table should have a primary key.

A table may have more than one combination of columns that could uniquely identify the rows in a table; each combination is a candidate key. The database administrator picks one of the candidate keys to be the primary key. For example, in the **part\_sample** table both **part\_nmbr** and **part\_name** could be candidate keys, but only **part\_nmbr** is chosen as a primary key.

CREATE TABLE part\_sample

1	— I	
(part_nmbr	int	PRIMARY KEY,
part_name	char(30),	
part_weight	decimal	(6,2),
part_color	char(15) )	

• FOREIGN KEY constraints identify the relationships between tables.

A foreign key in one table points to a candidate key in another table. Foreign keys prevent actions that would leave rows with foreign key values when there are no candidate keys with that value. In the following sample, the **order\_part** table establishes a foreign key referencing the **part\_sample** table defined earlier. Usually, **order\_part** would also have a foreign key against an order table, but this is a simple example.

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

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

- NO ACTION specifies that the deletion fails with an error.
- CASCADE specifies that all the rows with foreign keys pointing to the deleted row are also deleted.

The ON UPDATE clause defines the actions that are taken if you attempt to update a candidate key value to which existing foreign keys point. It also supports the NO ACTION and CASCADE options.

#### **Column and Table Constraints**

Constraints can be column constraints or table constraints:

- A column constraint is specified as part of a column definition and applies only to that column (the constraints in the earlier samples are column constraints).
- A table constraint is declared independently from a column definition and can apply to more than one column in a table.

Table constraints must be used when more than one column must be included in a constraint.

For example, if a table has two or more columns in the primary key, you must

use a table constraint to include both columns in the primary key. Consider a table that records events happening in a computer in a factory. Assume that events of several types can happen at the same time, but that no two events happening at the same time can be of the same type. This can be enforced in the table by including both the **type** and **time** columns in a two-column primary key.

CREATE TABLE factory\_process

(event\_type int, event\_time datetime, event\_site char(50), event\_desc char(1024), CONSTRAINT event\_key PRIMARY KEY (event\_type, event\_time) )

See Also

CREATE TABLE

Creating and Modifying a Table

## Rules

Rules are a backward-compatibility feature that perform some of the same functions as CHECK constraints. CHECK constraints are the preferred, standard way to restrict the values in a column. CHECK constraints are also more concise than rules; there can only be one rule applied to a column, but multiple CHECK constraints can be applied. CHECK constraints are specified as part of the CREATE TABLE statement, while rules are created as separate objects and then bound to the column.

This example creates a rule that performs the same function as the CHECK constraint example in the preceding topic. The CHECK constraint is the preferred method to use in Microsoft® SQL Server<sup>™</sup> 2000.

```
CREATE RULE id chk AS @id BETWEEN 0 and 10000
GO
CREATE TABLE cust_sample
 (
 cust id
              int
 PRIMARY KEY,
                char(50),
 cust name
                  char(50),
 cust address
 cust_credit_limit money,
 )
GO
sp_bindrule id_chk, 'cust_sample.cust_id'
GO
```

#### See Also

CREATE TABLE

Creating and Modifying a Table

## Defaults

Defaults specify what values are used in a column if you do not specify a value for the column when inserting a row. Defaults can be anything that evaluates to a constant, such as:

- Constant
- Built-in function
- Mathematical expression

There are two ways to apply defaults:

• Create a default definition using the DEFAULT keyword in CREATE TABLE to assign a constant expression as a default on a column.

This is the preferred, standard method. It is also the more concise way to specify a default.

• Create a default object using the CREATE DEFAULT statement and bind it to columns using the **sp\_bindefault** system stored procedure.

This is a backward compatibility feature.

This example creates a table using one of each type of default. It creates a default object to assign a default to one column, and binds the default object to the column. It then does a test insert without specifying values for the columns with defaults and retrieves the test row to verify the defaults were applied.

```
USE pubs
GO
CREATE TABLE test_defaults
(keycol smallint,
process_id smallint DEFAULT @@SPID, --Preferred default defin
date_ins datetime DEFAULT getdate(), --Preferred default definiti
```

smallint DEFAULT 10 \* 2, --Preferred default definition mathcol char1 char(3), char(3) DEFAULT 'xyz') -- Preferred default definition char2 GO /\* Illustration only, use DEFAULT definitions instead.\*/ CREATE DEFAULT abc\_const AS 'abc' GO sp\_bindefault abc\_const, 'test\_defaults.char1' GO INSERT INTO test\_defaults(keycol) VALUES (1) GO SELECT \* FROM test\_defaults GO

The output of this sample is:

Default bound to column.

(1 row(s) affected)

keycol process\_id date\_insmathcol char1 char217Oct 16 1997 8:34PM20abcxyz

(1 row(s) affected)

#### See Also

CREATE TABLE

Creating and Modifying a Table

# Triggers

Microsoft® SQL Server<sup>™</sup> 2000 triggers are a special class of stored procedure defined to execute automatically when an UPDATE, INSERT, or DELETE statement is issued against a table or view. Triggers are powerful tools that sites can use to enforce their business rules automatically when data is modified. Triggers can extend the integrity checking logic of SQL Server constraints, defaults, and rules, although constraints and defaults should be used instead whenever they provide all the needed functionality.

Tables can have multiple triggers. The CREATE TRIGGER statement can be defined with the FOR UPDATE, FOR INSERT, or FOR DELETE clauses to target a trigger to a specific class of data modification actions. When FOR UPDATE is specified, the IF UPDATE (*column\_name*) clause can be used to target a trigger to updates affecting a particular column.

Triggers can automate the processing for a company. In an inventory system, update triggers can detect when a stock level reaches a reorder point and generate an order to the supplier automatically. In a database recording the processes in a factory, triggers can e-mail or page operators when a process exceeds defined safety limits.

The following trigger generates an e-mail whenever a new title is added in the **pubs** database:

CREATE TRIGGER reminder ON titles FOR INSERT AS EXEC master..xp\_sendmail 'MaryM', 'New title, mention in the next report to distributors.'

Triggers contain Transact-SQL statements, much the same as stored procedures. Triggers, like stored procedures, return the result set generated by any SELECT statements in the trigger. Including SELECT statements in triggers, except statements that only fill parameters, is not recommended. This is because users do not expect to see any result sets returned by an UPDATE, INSERT, or DELETE statement.

You can use the FOR clause to specify when a trigger is executed:

• AFTER

The trigger executes after the statement that triggered it completes. If the statement fails with an error, such as a constraint violation or syntax error, the trigger is not executed. AFTER triggers cannot be specified for views, they can only be specified for tables. You can specify multiple AFTER triggers for each triggering action (INSERT, UPDATE, or DELETE). If you have multiple AFTER triggers for a table, you can use **sp\_settriggerorder** to define which AFTER trigger fires first and which fires last. All other AFTER triggers besides the first and last fire in an undefined order which you cannot control.

AFTER is the default in SQL Server 2000. You could not specify AFTER or INSTEAD OF in SQL Server version 7.0 or earlier, all triggers in those versions operated as AFTER triggers.

• INSTEAD OF

The trigger executes in place of the triggering action. INSTEAD OF triggers can be specified on both tables and views. You can define only one INSTEAD OF trigger for each triggering action (INSERT, UPDATE, and DELETE). INSTEAD OF triggers can be used to perform enhance integrity checks on the data values supplied in INSERT and UPDATE statements. INSTEAD OF triggers also let you specify actions that allow views, which would normally not support updates, to be updatable.

#### See Also

Enforcing Business Rules with Triggers

SQL Server Setup Help

# Collations

The physical storage of character strings in Microsoft® SQL Server<sup>™</sup> 2000 is controlled by collations. A collation specifies the bit patterns that represent each character and the rules by which characters are sorted and compared.

SQL Server 2000 supports objects that have different collations being stored in a single database. Separate SQL Server 2000 collations can be specified down to the level of columns. Each column in a table can be assigned different collations. Earlier versions of SQL Server support only one collation for each instance of SQL Server. All databases and database objects created in an instance of SQL Server 7.0 or earlier have the same collation.

## How Character Data Is Stored

In a computer, characters are represented by different patterns of bits being either ON or OFF. There are 8 bits in a byte, and the 8 bits can be turned ON and OFF in 256 different patterns. A program that uses 1 byte to store each character can therefore represent up to 256 different characters by assigning a character to each of the bit patterns. There are 16 bits in 2 bytes, and 16 bits can be turned ON and OFF in 65,536 unique patterns. A program that uses 2 bytes to represent each character can represent up to 65,536 characters.

Single-byte code pages are definitions of the characters mapped to each of the 256 bit patterns possible in a byte. Code pages define bit patterns for uppercase and lowercase characters, digits, symbols, and special characters such as !, @, #, or %. Each European language, such as German or Spanish, has its own single-byte code page. Although the bit patterns used to represent the Latin alphabet characters A through Z are the same for all the code pages, the bit patterns used to represent accented characters such as 'é' and 'á' vary from one code page to the next. If data is exchanged between computers running different code pages, all character data must be converted from the code page of the sending computer to the code page of the receiving computer. If the source data has extended characters that are not defined in the code page of the receiving computer, data is lost. When a database serves clients from many different countries, it is difficult to pick a code page for the database that contains all the extended characters required by all the client computers. Also, there is a lot of processing time spent

doing the constant conversions from one code page to another.

Single-byte character sets are also inadequate to store all the characters used by many languages. For example, some Asian languages have thousands of characters, so must use two bytes per character. Double-byte character sets have been defined for these languages. Still, each of these languages have their own code page, and there are difficulties in transferring data from a computer running one double-byte code page to a computer running another.

Code page	Description
1258	Vietnamese
1257	Baltic
1256	Arabic
1255	Hebrew
1254	Turkish
1253	Greek
1252	Latin1 (ANSI)
1251	Cyrillic
1250	Central European
950	Chinese (Traditional)
949	Korean
936	Chinese (Simplified)
932	Japanese
874	Thai
850	Multilingual (MS-DOS Latin1)
437	MS-DOS U.S. English

SQL Server 2000 supports these code pages.

To address the character conversion and interpretation problems that occur when trying to support multiple code pages in a network, the ISO standards organization and a group called the Unicode Consortium defined the Unicode standard. Unicode uses two bytes to store each character. Because 65,536 characters are enough to cover all the commonly used characters from all the languages of the world, all major languages are covered by the Unicode standard. If all the computers and programs in a network use Unicode, there is no need for any character conversions, each user will see exactly the same

characters as all other users, and no loss of characters will occur.

On computers running Microsoft Windows® operating systems, the code page used by the operating system and Windows applications is defined by the Windows locale. The locale is selected when the operating system is installed. Windows applications interpret character data using the code page defined by the Windows locale. Windows applications also support wide character, or Unicode, data.

SQL Server 2000 supports two categories of character data types:

- The Unicode data types **nchar**, **nvarchar**, and **ntext**. These data types use the Unicode character representation. Code pages do not apply to these data types.
- The non-Unicode character data types **char**, **varchar**, and **text**. These data types use the character representation scheme defined in a single or double-byte code page.

For more information about how character data is stored and the operation of code pages, Unicode, and sort orders, see Developing International Software for Windows 95 and Windows NT 4.0 in the MSDN® page at <a href="http://msdn.microsoft.com">http://msdn.microsoft.com</a>.

## International Data and Unicode

Storing data in multiple languages within one database is difficult to manage when using only character data and code pages. It is difficult to find one code page for the database that can store all the required language-specific characters. It is also difficult to ensure the proper translation of special characters when being read or updated by different clients running various code pages. Databases that support international clients should always use Unicode data types instead of non-Unicode data types.

For example, a database of customers in North America has to handle three major languages:

• Spanish names and addresses for Mexico.

- French names and addresses for Quebec.
- English names and addresses for the rest of Canada and the United States.

When you use only character columns and code pages, care has to be taken to ensure the database is installed with a code page that will handle the characters of all three languages. More care must be taken to ensure the proper translation of characters from one of the languages when read by clients running a code page for another language.

With the growth of the Internet, it is becoming more important than ever before to support many client computers running different locales. It is difficult to pick a code page for character data types that will support all of the characters required by a worldwide audience.

The easiest way to manage character data in international databases is to always use the Unicode **nchar**, **nvarchar**, and **ntext** data types in place of their non-Unicode equivalents (**char**, **varchar**, and **text**). If all the applications that work with international databases also use Unicode variables instead of non-Unicode variables, character translations do not have to be performed anywhere in the system. All clients will see exactly the same characters in data as all other clients.

For systems that could use single-byte code pages, the fact that Unicode data needs twice as much storage space as non-Unicode character data is at least partially offset by eliminating the need to convert extended characters between code pages. Systems using double-byte code pages do not have this issue.

SQL Server 2000 stores all textual system catalog data in columns having Unicode data types. The names of database objects such as tables, views, and stored procedures are stored in Unicode columns. This allows applications to be developed using only Unicode, which avoids all issues with code page conversions.

## Sort Order

A sort order specifies the rules used by SQL Server to interpret, collate, compare, and present character data. For example, a sort order defines whether 'a' is less

than, equal to, or greater than 'b'. A sort order defines whether the collation is case-sensitive, for example whether 'm' is equal or not equal to 'M'. It also defines if the collation is accent-sensitive, for example whether 'á' is equal or not equal to 'ä'.

SQL Server 2000 uses two sort orders with each collation, one for Unicode data and another for the character code page.

Many SQL Server collations use the same code page, but have a different sort order for the code page. This allows sites to choose:

- Whether characters will simply be sorted based on the numeric value represented by their bit patterns. Binary sorting is fastest because SQL Server does not have to make any adjustments and can use fast, simple sorting algorithms. Binary sort orders are always case-sensitive. Because the bit patterns in a code page may not be arranged in the same sequence as defined by the dictionary rules for a specific language, binary sorting sometimes does not sort characters in a sequence users who speak that language might expect.
- Between case-sensitive or case-insensitive behavior.
- Between accent-sensitive or accent-insensitive behavior.

#### See Also

Collation Options for International Support SQL Server Collation Fundamentals Unicode Data Using Unicode Data

# **SQL Server Collation Fundamentals**

Microsoft® SQL Server<sup>™</sup> 2000 supports several collations. A collation encodes the rules governing the proper use of characters for either a language, such as Macedonian or Polish, or an alphabet, such as Latin1\_General (the Latin alphabet used by western European languages).

Each SQL Server collation specifies three properties:

- The sort order to use for Unicode data types (**nchar**, **nvarchar**, and **ntext**). A sort order defines the sequence in which characters are sorted, and the way characters are evaluated in comparison operations.
- The sort order to use for non-Unicode character data types (char, varchar, and text).
- The code page used to store non-Unicode character data.

**Note** You cannot specify the equivalent of a code page for the Unicode data types (**nchar**, **nvarchar**, and **ntext**). The double-byte bit patterns used for Unicode characters are defined by the Unicode standard and cannot be changed.

SQL Server 2000 collations can be specified at any level. When you install an instance of SQL Server 2000, you specify the default collation for that instance. Each time you create a database, you can specify the default collation used for the database. If you do not specify a collation, the default collation for the database is the default collation for the instance. Whenever you define a character column, variable, or parameter, you can specify the collation of the object. If you do not specify a collation, the object is created with the default collation of the database.

If all of the users of your instance of SQL Server speak the same language, you should pick the collation that supports that language. For example, if all of the users speak French, choose the French collation.

If the users of your instance of SQL Server speak multiple languages, you should pick a collation that best supports the requirements of the various languages. For example, if the users generally speak western European languages, choose the Latin1\_General collation. When you support users who speak multiple languages, it is most important to use the Unicode data types, **nchar**, **nvarchar**, and **ntext**, for all character data. Unicode was designed to eliminate the code page conversion difficulties of the non-Unicode **char**, **varchar**, and **text** data types. Collation still makes a difference when you implement all columns using Unicode data types because it defines the sort order for comparisons and sorts of Unicode characters. Even when you store your character data using Unicode data types you should pick a collation that supports most of the users in case a column or variable is implemented using the non-Unicode data types.

A SQL Server collation defines how the database engine stores and operates on character and Unicode data. After data has been moved into an application, however, character sorts and comparisons done in the application are controlled by the Windows locale selected on the computer. The collation used for character data by applications is one of the items controlled by the Windows locale also defines other items, such as number, time, date, and currency formats). For Microsoft Windows NT® 4.0, Microsoft Windows® 98, and Microsoft Windows 95, the Windows locale is specified using the Regional Settings application in Control Panel. For Microsoft Windows 2000, the locale is specified using the Regional Options application in Control Panel. For more information about Windows NT 4.0 in the MSDN® page at Microsoft Web site.

Multiple collations can use the same code page for non-Unicode data. For example, the 1251 code page defines a set of Cyrillic characters. This code page is used by several collations, such as Cyrillic\_General, Ukrainian, and Macedonian. Although all of these collations use the same set of bits to represent non-Unicode character data, the sorting and comparison rules they apply are slightly different to handle the dictionary definitions of the correct sequence of characters in the language or alphabet associated with the collation.

Because SQL Server 2000 collations control both the Unicode and non-Unicode sort orders, you do not encounter problems caused by specifying different sorting rules for Unicode and non-Unicode data. In earlier versions of SQL Server, the code page number, the character sort order, and the Unicode collation are specified separately. Earlier versions of SQL Server also support varying numbers of sort orders for each code pages, and for some code pages support sort orders not available in Windows locales. In SQL Server 7.0, it is also possible to specify a Unicode sort order that is different from the sort order chosen for non-Unicode data. This can cause ordering and comparison operations to return different results when working with Unicode data as opposed to non-Unicode data.

#### See Also

COLLATE Collation Options for International Support Collations Unicode Data Using Unicode Data SQL Server Setup Help

# **Selecting Collations**

In Microsoft® SQL Server<sup>™</sup> 2000, you specify a single collation name that controls all three collation attributes: the Unicode sort order, the non-Unicode code page, and the non-Unicode sort order. None of the SQL Server 2000 collations allow different comparison and sorting rules for Unicode and non-Unicode character data. There are two groups of SQL Server 2000 collations: Windows collations and SQL collations.

## Windows Collations

Windows collations are collations defined for SQL Server to support Microsoft Windows® locales. By specifying a Windows collation for SQL Server, the instance of SQL Server uses the same code pages and sorting and comparison rules as an application running on a computer for which you have specified the associated Windows locale. For example, the French Windows collation for SQL Server matches the collation attributes of the French locale for Windows.

There are more Windows locales than there are SQL Server Windows collations. The names of Windows locales are based on a language and territory, for example French (Canada). Several languages, however, share common alphabets and rules for sorting and comparing characters. For example, 33 Windows locales, including all of the Portuguese, and English Windows locales, use the Latin1 code page (1252) and follow a common set of rules for sorting and comparing characters. The SQL Server Windows collation based on the Latin1\_General code page and sorting rules supports all 33 of these Windows locales. Also, Windows locales specify attributes not covered by SQL Server Windows collations, such as currency, date, and time formats. Because countries such as Great Britain and the United States have different currency, date, and time formats, they require different Windows collations. They do not require different SQL Server collations because they have the same alphabet and rules for sorting and comparing characters.

## **SQL Collations**

SQL collations are a compatibility option to match the attributes of common

combinations of code page number and sort orders that have been specified in earlier versions of SQL Server. For example, for mapping a SQL Server 2000 SQL collation to what is specified in earlier versions of SQL Server, the SQL Server 2000 SQL collation SQL\_Latin1\_General\_CP1\_CI\_AS matches the SQL Server version 7.0 default specification of:

- The ISO code page 1252.
- The dictionary order, case-insensitive character sort order.
- The General Unicode collation.

The SQL collations available in SQL Server 2000 do not match all combinations that can be specified in earlier versions of SQL Server. For example, no SQL Server 2000 SQL collation supports a case-sensitive sort order for non-Unicode data and case-insensitive sort order for Unicode data. The earlier SQL collations that cannot be exactly specified in SQL Server 2000 are called obsolescent SQL collations.

In SQL Server 2000, you should primarily use Windows collations. You should use SQL collations only to maintain compatibility with existing instances of earlier versions of SQL Server, or to maintain compatibility in applications developed using SQL collations in earlier versions of SQL Server.

## **Collation Comparison and Ordering Rules**

Most of the comparison and ordering rules defined in a collation are governed by the dictionary definition of the correct sequence of characters for the alphabet or language. The attributes you can control are whether comparisons and sorts of character and Unicode data should be:

- Based on the dictionary conventions that define the correct sequence of characters in the language or alphabet associated with the collation, or based on the sequence of the binary bit patterns representing the different characters.
- Case-sensitive or case-insensitive. For example, defining whether 'a' is

equal or not equal to 'A'. If you choose case-insensitive, comparisons always ignore case, so the uppercase version of a character evaluates to being equal to the lowercase version of the character. When you choose case-insensitivity, the relative sequence in which uppercase and lowercase are sorted is undefined unless you also specify uppercase preference. Uppercase preference affects only sort operations and specifies that uppercase versions of a character come earlier in the sort sequence than lowercase versions of the same character. Uppercase preference has no affect on comparisons, so 'A' still evaluates to being equal to 'a' when uppercase preference is on. Uppercase preference can be specified only in SQL collations, not in Windows collations.

 Sensitive or insensitive to accented characters, also known as extended characters. Accented characters are those characters that have a diacritical mark, such as the German umlaut (ë) or the Spanish tilde (~). For example, accent sensitivity defines whether 'a' is equal or not equal to 'ä'.

When you choose a collation, you can specify if you want binary behavior, or dictionary sorting that is sensitive or insensitive to case and accents:

In binary collations, comparisons and sorting are based strictly on the bit pattern of the characters. This is the fastest option. Because uppercase characters are stored with different bit patterns than their corresponding lowercase characters, and accented characters have different bit patterns than characters without accents, binary sort orders are always case-sensitive and accent sensitive. Binary collations also ignore dictionary sequences that have been defined for specific languages. They simply order the characters based on the relative value of the bit patterns that represent each character. While the bit patterns defined for Latin characters, such as 'A' or 'z', are such that binary sorting yields the correct results, the bit patterns for some extended characters in some code pages may be different than the ordering sequence defined in dictionaries for the language associated with a collation. This can lead to occasional ordering and comparison results that are different than what a speaker of the language might expect.

• If you do not specify a binary collation, SQL Server uses the dictionary ordering of the collation you have chosen. Dictionary order means characters are not sorted or compared based only on their bit patterns. The collation follows the conventions of the associated language regarding the proper sequence for characters. For example, case-insensitive sort orders must use dictionary rules to determine which lowercase and uppercase bit patterns are equal.

Although the bit patterns in a code page generally yield the correct comparison and ordering results for any language that uses the code page, the conventions for some of the languages may require different results than are generated for the bit patterns of a small number of characters. For example, the Czech, Hungarian, and Polish collations use the same code page, 1250, which was designed for the Slavic languages. Each of these languages, however, use slightly different conventions for the sequence in which accented characters should be sorted.

If you do not specify binary sorting, all SQL Server operations follow the dictionary conventions for sorting and comparing characters. When the dictionary order is used, you can specify whether you want the collation to be sensitive or insensitive to both case and accented characters.

Case-sensitivity applies to SQL identifiers and passwords as well as to data. If you specify a binary or case-sensitive default sort order for an instance of SQL Server or database, all references to objects must use the same case with which they were created. For example, consider this table:

#### CREATE TABLE MyTable (PrimaryKey int PRIMARY KEY, CharCo

If the CREATE TABLE statement is executed on an instance of SQL Server or database that has a case-sensitive or binary sort order, all references to the table must use the same case that was specified in the CREATE TABLE statement:

-- Object not found error because case is not correct: SELECT \* FROM MYTABLE

- -- Invalid column name error because case is not correct
- -- for the WHERE clause reference to the PrimaryKey column.

SELECT \* FROM MyTable WHERE PRIMARYKEY = 123 -- Correct statement: SELECT CharColumn FROM MyTable WHERE PrimaryKey = 123

#### See Also

Collation Options for International Support Specifying Collations Unicode Data Using Unicode Data

# **Specifying Collations**

Microsoft<sup>®</sup> SQL Server<sup>™</sup> 2000 collations can be specified at several levels, including the following:

- When you install an instance of SQL Server, you can specify the default collation for that instance during setup. The default collation for the instance also becomes the default collation of the system databases: **master**, **model**, **tempdb**, **msdb**, and **Distribution**.
- When you create a database, you can use the COLLATE clause of the CREATE DATABASE statement to specify the default collation of the database. You can also specify a collation when you create a database using SQL Server Enterprise Manager. If you do not specify a collation, the database is assigned the default collation of the **model** database. The default collation of the **model** database is the same as the default collation of the instance of SQL Server.
- When you create a table, you can specify collations for each character string column using the COLLATE clause of the CREATE TABLE statement. You can also specify a collation when you create a table using SQL Server Enterprise Manager. If you do not specify a collation, the column is assigned the default collation of the database.

You can also use the database\_default option in the COLLATE clause to specify that a column in a temporary table use the collation default of the current user database for the connection instead of **tempdb**.

- When you specify a literal string, you can use the COLLATE clause to specify the collation. If you do not specify a collation, the literal is assigned the database default collation.
- In SQL-DMO you can use the **Collation** property to specify collations for instances, databases, and columns. For more information, see

#### Collation Property.

• Parameters for stored procedures or functions, user-defined data types, and variables are assigned the default collation of the database:

The collation of an identifier depends on the level at which it is defined. Identifiers of instance-level objects, such as logins and database names, are assigned the default collation of the instance. Identifiers of objects within a database, such as tables, views, and column names, are assigned the default collation of the database. Variables, GOTO labels, temporary stored procedures, and temporary tables can be created when the connection context is associated with one database, and then referenced when the context has been switched to another database. Because of this, the identifiers for variables, GOTO labels, and temporary tables are in the default collation of the instance.

Specifying collations for columns or literals can be done only for the **char**, **varchar**, **text**, **nchar**, **nvarchar**, and **ntext** data types.

Collations are generally identified by a collation name. There are two classes of names: Windows collation names for the new collations aligned with Windows locales, and SQL collation names for the compatibility mode collations that result when upgrading from earlier versions of SQL Server. For more information, see <u>Windows Collation Name</u>), and <u>SQL Collation Name</u>.

The exception to specifying collation names is in Setup:

- You do not specify a collation name for Windows collations, but instead specify the collation designator, and then select check boxes to specify binary sorting or dictionary sorting that is either sensitive or insensitive to either case or accents.
- You do not specify SQL collation names, but instead select a collation based on a longer, more human-readable display name.

You can execute the system function **fn\_helpcollations** to retrieve a list of all the valid collation names for Windows collations and SQL collations, for example:

#### SELECT \*

## FROM ::fn\_helpcollations()

You can also use the SQL-DMO **ListCollations** method to get a list of the valid collation names. For more information, see <u>ListCollations Method</u>.

The system catalog stored procedures have been enhanced to report the collation of all SQL Server objects that have a collation.

SQL Server can support only code pages that are supported by the underlying operating system. When you perform an action that depends on collations, the SQL Server collation used by the referenced object must use a code page supported by the operating system running on the computer. These actions can include:

- Specifying a default collation for an instance of SQL Server.
- Specifying a default collation for a database when you create the database.
- Restoring a database backup. Windows must support the code page of the default collation used by the database.
- Attaching a database. Windows must support the code page of the default collation used by the database.
- Specifying a collation for a column when creating a table.
- Specifying a collation when creating a user-defined data type.
- Specifying a collation when declaring a character-string constant.

If the collation specified or the collation used by the referenced object, uses a code page not supported by the Microsoft Windows® operating systems, SQL Server issues error 2775:

"Code page *codepagenumber* is not supported by the system."

Your response to this message depends on the version of the Windows operating system installed on the computer:

- Microsoft Windows 2000 supports all of the code pages used by SQL Server collations, so the error message will not occur.
- Microsoft Windows NT® 4.0 may require that you install a language pack to support some code pages. For more information about installing a Windows NT language pack, see the Windows NT Help.
- Microsoft Windows 98 supports only one code page on a computer. You must choose a SQL Server collation that uses the same code page used by Windows 98.

#### See Also

ALTER TABLE Collation Options for International Support Collations Constants CREATE DATABASE CREATE TABLE DECLARE @local\_variable table

Using Unicode Data

# Specifying the Default Collation for an Instance of SQL Server

The default collation for an instance of Microsoft® SQL Server<sup>™</sup> 2000 is defined during setup. If you choose the minimal or typical setup options, then Setup installs these collations:

- If you upgrade a default instance of SQL Server version 6.5 or SQL Server version 7.0 to SQL Server 2000, or if you install a default instance of SQL Server 2000 that will be version switched with a default instance of SQL Server version 6.5, SQL Server Setup carries forward the same collation used in the existing instance of SQL Server version 6.5 or SQL Server version 7.0, including obsolescent collations.
- In all other cases, Setup chooses the Windows collation that supports the Windows locale of the computer on which the instance of SQL Server 2000 is being installed.

**Note** The Setup program does not set the instance default collation to the Windows collation Latin1\_General\_CI\_AS if the computer is using the U.S. English locale. Instead, it sets the instance default collation to the SQL collation SQL\_Latin1\_General\_Cp1\_CI\_AS. This may change in a future release.

If you choose the **Custom** setup option, Setup uses the same logic as in the minimal and typical options to set the collation that is selected when the **Character Set / Sort Order / Windows Collation** window is displayed. You should not use the selected collation in these cases:

- If the instance will be included in a replication scheme, all instances of SQL Server involved in the replication scheme (Publishers, Subscribers, and Distributors) should use the same code page. You should make sure the collation selected by Setup uses the same code page as the other instances of SQL Server in the replication scheme.
- If the primary language that the instance must support is different than

the Windows locale of the computer on which the instance is being installed.

For a table showing which collation designator to specify for a Windows locale, see <u>Windows Collation Names Table</u>.

During setup, the **master**, **model**, **tempdb**, **msdb**, and **Distribution** system databases are assigned the same default collation as the default collation chosen for the instance.

#### See Also

**Collation Options for International Support** 

SQL Server Setup Help

# **Mixed Collation Environments**

Compatibility issues can have an impact on organizations that use multiple collations to store their data. Most organizations use the same collation for all of their Microsoft® SQL Server<sup>™</sup> 2000 databases, thereby eliminating all collation compatibility issues. Other organizations, however, must store data viewed by users who speak various languages and want to do so with a minimum of collation compatibility issues.

All character and Unicode objects (such as columns, variables, and constants) have a collation. Whenever you work with objects that have different collations and code pages, you must code your queries to comply with the rules of collation coercion. When you code a complex expression that uses operators to combine multiple simple expressions that have different collations, all of the collations must be implicitly convertible, or explicitly converted using the COLLATE clause. For more information about collation coercion, see <u>Collation Precedence</u>.

If you do not specify a collation in a character or Unicode expression, the default collation may vary depending on the current database setting for the connection. For example, if you do not specify a COLLATE clause on a character or Unicode constant, the constant is assigned the default collation of the current database. This means that the result of a Transact-SQL statement may have different collations when executed in the context of different databases.

If you are setting up replication, all of the databases involved in a replication network, including Publishers, Subscribers, and Distributors, must have the same code page.

The bulk copy functions, BULK INSERT, and the **bcp** command prompt utility support column collations. For more information, see <u>Copying Data Between</u> <u>Different Collations</u>.

#### **Minimizing Collation Issues**

If you must store character data that reflects multiple languages, you can minimize collation compatibility issues by always using the Unicode **nchar**, **nvarchar**, and **ntext** data types instead of the **char**, **varchar**, **text** data types.

Using the Unicode data types eliminates code page conversion issues.

Another recommendation that minimizes collation compatibility issues is to standardize your site as either sensitive or insensitive to case and accented characters. If you always choose collations with the same case and accent sensitivity, end users experience consistent behavior across all systems. Most SQL Server 2000 sites choose to be case-insensitive and accent-sensitive. Case sensitivity also applies to the names of SQL Server objects; therefore, if you specify case-sensitive collations, all users must specify the correct case when querying the database. For example, if you have a case-sensitive server and create a table named **Employees**, all queries must refer to the table as **Employees**. References that do not use the correct case, such as **EMPLOYEES** or **employees**, are invalid.

## **Collations and tempdb**

The **tempdb** database is built each time SQL Server is started, and has the same default collation as the **model** database, which is typically the same as the default collation of the instance. If you create a user database and specify a different default collation than **model**, the user database has a different default collation than **model**, the user database has a different default collation than **tempdb**. All temporary stored procedures or temporary tables are created and stored in **tempdb**, which means that all implicit columns in temporary tables and all coercible-default constants, variables, and parameters in temporary stored procedures have different collations than comparable objects created in permanent tables and stored procedures.

This can lead to problems with the **text** data type, which does not support code page conversions. For example, an instance of SQL Server 2000 defaults to the Latin1\_General\_CS\_AS collation, and you execute these statements:

CREATE DATABASE TestDB COLLATE Estonian\_CS\_AS USE TestDB CREATE TABLE TestPermTab (PrimaryKey int PRIMARY KEY, Tex

In this system, the **tempdb** database uses the Latin1\_General\_CS\_AS collation with code page 1252, and **TestDB** and **TestPermTab.TextCol** use the Estonian\_CS\_AS collation with code page 1257. If you then execute:

USE TestDB

GO

-- Create a temporary table with the same column declarations

-- as TestPermTab

CREATE TABLE #TestTempTab (PrimaryKey int PRIMARY KEY, Te

-- This statement gets an code page conversion not allowed error

-- because the temporary table is created in tempdb, which has a -- different default collation than TestDB.

INSERT INTO #TestTempTab

```
SELECT * FROM TestPermTab
```

GO

To eliminate the error you can use one of these alternatives:

- Use the Unicode data type **ntext** instead of **text** for the two **TextCol** columns.
- Specify that the temporary table column use the default collation of the user database, not **tempdb**. This allows the temporary table to work with similarly formatted tables in multiple databases, if that is a requirement of your system.

CREATE TABLE #TestTempTab (PrimaryKey int PRIMARY KEY, TextCol text COLLATE database\_default )

 Specify the correct collation for the #TestTempTab column: CREATE TABLE #TestTempTab (PrimaryKey int PRIMARY KEY, TextCol text COLLATE Estonian\_CS\_AS )

## **Collations in BACKUP and RESTORE**

If you restore a database, RESTORE uses the collation of the source database that was recorded in the backup file. The restored database has the same

collation as the original database that was backed up. Individual objects within the database that have different collations also retain their original collation. The database can be restored even if the instance on which you run restore has a different default collation than the instance on which BACKUP was run.

If there is already a database with the same name on the target server, the only way to restore from the backup is to specify REPLACE on the RESTORE statement. If you specify REPLACE, the existing database is completely replaced with the contents of the database on the backup file, and the restored version of the database will have the same collation recorded in the backup file.

If you are restoring log backups, the destination database must have the same collation as the source database.

## **Collations and text column**

If you create a table with a **text** column that has a different code page than the code page of the database's default collation, there are only two ways you can specify data values to be inserted into the column, or update existing values. You can:

- Specify a Unicode constant.
- Select a value from another column with the same code page.

Assume the following database and table:

 -- Create a database with a default of code page 1252.
 CREATE DATABASE TestDB COLLATE Latin1\_General\_CS\_AS
 -- Create a table with a different code page, 1253.
 CREATE TABLE TestTab

 (PrimaryKey int PRIMARY KEY, TextCol text COLLATE Greek\_CS\_AS
 )

-- This INSERT statement successfully inserts a Unicode string. INSERT INTO TestTab VALUES (1, N'abc')

-- This INSERT statement successfully inserts data by selecting

-- from a similarly formatted table in another database that uses

-- uses the Greek 1253 code page as its default.

INSERT INTO TestTab

SELECT \* FROM GreekDatabase.dbo.TestTab

SQL Server Setup Help

# **Changing Collations**

You can change the collation of a column by using the ALTER TABLE statement:

```
CREATE TABLE MyTable
(PrimaryKey int PRIMARY KEY,
CharCol varchar(10) COLLATE French_CI_AS NOT NULL
)
GO
ALTER TABLE MyTable ALTER COLUMN CharCol
varchar(10)COLLATE Latin1_General_CI_AS NOT NULL
GO
```

You cannot alter the collation of a column that is currently referenced by:

- A computed column.
- An index.
- Distribution statistics, either generated automatically or by the CREATE STATISTICS statement.
- A CHECK constraint.
- A FOREIGN KEY constraint.

You can also use the COLLATE clause on an ALTER DATABASE to change the default collation of the database:

ALTER DATABASE MyDatabase COLLATE French\_CI\_AS

Altering the default collation of a database does not change the collations of the

columns in any existing user-defined tables. These can be changed with ALTER TABLE. The COLLATE CLAUSE on an ALTER DATABASE statement changes:

- The default collation for the database. This new default collation is applied to all columns, user-defined data types, variables, and parameters subsequently created in the database. It is also used when resolving the object identifiers specified in SQL statements against the objects defined in the database.
- Any **char**, **varchar**, **text**, **nchar**, **nvarchar**, or **ntext** columns in system tables to the new collation.
- All existing **char**, **varchar**, **text**, **nchar**, **nvarchar**, or **ntext** parameters and scalar return values for stored procedures and user-defined functions to the new collation.
- The **char**, **varchar**, **text**, **nchar**, **nvarchar**, or **ntext** system data types, and all user-defined data types based on these system data types, to the new default collation.

After a collation has been assigned to any object other than a column or database, you cannot change the collation except by dropping and re-creating the object. This can be a complex operation. To change the default collation for an instance of Microsoft® SQL Server<sup>TM</sup> 2000 you must:

- Make sure you have all of the information or scripts needed to re-create your user databases and all of the objects in them.
- Export all of your data using a tool such as bulk copy.
- Drop all of the user databases.
- Rebuild the **master** database specifying the new collation.

- Create all of the databases and all of the objects in them.
- Import all of your data.

**Note** Instead of changing the default collation of an instance of SQL Server 2000, you can specify a default collation for each new database you create.

SQL Server Setup Help

# **SQL Indexes**

A Microsoft® SQL Server<sup>™</sup> 2000 index is a structure associated with a table or view that speeds retrieval of rows from the table or view. An index contains keys built from one or more columns in the table or view. These keys are stored in a structure that allows SQL Server to find the row or rows associated with the key values quickly and efficiently.

SQL Server Setup Help

# **Table Indexes**

Microsoft<sup>®</sup> SQL Server<sup>™</sup> 2000 supports indexes defined on any column in a table, including computed columns.

If a table is created with no indexes, the data rows are not stored in any particular order. This structure is called a heap.

The two types of SQL Server indexes are:

• Clustered

Clustered indexes sort and store the data rows in the table based on their key values. Because the data rows are stored in sorted order on the clustered index key, clustered indexes are efficient for finding rows. There can only be one clustered index per table, because the data rows themselves can only be sorted in one order. The data rows themselves form the lowest level of the clustered index.

The only time the data rows in a table are stored in sorted order is when the table contains a clustered index. If a table has no clustered index, its data rows are stored in a heap.

• Nonclustered

Nonclustered indexes have a structure completely separate from the data rows. The lowest rows of a nonclustered index contain the nonclustered index key values and each key value entry has pointers to the data rows containing the key value. The data rows are not stored in order based on the nonclustered key.

The pointer from an index row in a nonclustered index to a data row is called a row locator. The structure of the row locator depends on whether the data pages are stored in a heap or are clustered. For a heap, a row locator is a pointer to the row. For a table with a clustered index, the row locator is the clustered index key.

The only time the rows in a table are stored in any specific sequence is when a clustered index is created on the table. The rows are then stored in sequence on

the clustered index key. If a table only has nonclustered indexes, its data rows are stored in a unordered heap.

Indexes can be unique, which means no two rows can have the same value for the index key. Otherwise, the index is not unique and multiple rows can share the same key value.

There are two ways to define indexes in SQL Server. The CREATE INDEX statement creates and names an index. The CREATE TABLE statement supports the following constraints that create indexes:

- PRIMARY KEY creates a unique index to enforce the primary key.
- UNIQUE creates a unique index.
- CLUSTERED creates a clustered index.
- NONCLUSTERED creates a nonclustered index.

When you create an index on SQL Server 2000, you can specify whether the keys are stored in ascending or descending order.

SQL Server 2000 supports indexes defined on computed columns, as long as the expression defined for the column meets certain restrictions, such as only referencing columns from the table containing the computed column, and being deterministic.

A fill factor is a property of a SQL Server index that controls how densely the index is packed when created. The default fill factor usually delivers good performance, but in some cases it may be beneficial to change the fill factor. If the table is going to have many updates and inserts, create an index with a low fill factor to leave more room for future keys. If the table is a read-only table that will not change, create the index with a high fill factor to reduce the physical size of the index, which lowers the number of disk reads SQL Server uses to navigate through the index. Fill factors are only applied when the index is created. As keys are inserted and deleted, the index will eventually stabilize at a certain density.

Indexes not only speed up the retrieval of rows for selects, they also usually increase the speed of updates and deletes. This is because SQL Server must first find a row before it can update or delete the row. The increased efficiency of using the index to locate the row usually offsets the extra overhead needed to update the indexes, unless the table has a lot of indexes.

This example shows the Transact-SQL syntax for creating indexes on a table.

```
USE pubs
GO
CREATE TABLE emp_sample
(emp_id int PRIMARY KEY CLUSTERED,
emp_name char(50),
emp_address char(50),
emp_title char(25) UNIQUE NONCLUSTERED )
GO
CREATE NONCLUSTERED INDEX sample_nonclust ON emp_samp
GO
```

Deciding which particular set of indexes will optimize performance depends on the mix of queries in the system. Consider the clustered index on **emp\_sample.emp\_id**. This works well if most queries referencing **emp\_sample** have equality or range comparisons on **emp\_id** in their WHERE clauses. If the WHERE clauses of most queries reference **emp\_name** instead of **emp\_id**, performance could be improved by instead making the index on **emp\_name** the clustered index.

Many applications have a complex mix of queries that is difficult to estimate by interviewing users and programmers. SQL Server 2000 provides an Index Tuning Wizard to help design indexes in a database. The easiest way to design indexes for large schemas with complex access patterns is to use the Index Tuning Wizard.

You provide the Index Tuning Wizard with a set of SQL statements. This could be a script of statements you build to reflect a typical mix of statements in the system, but it is usually a SQL Profiler trace of the actual SQL statements processed on the system during a period of time that reflects the typical load on the system. The Index Tuning Wizard analyzes the workload and the database, and then recommends an index configuration that will improve the performance of the workload. You can choose to either replace the existing index configuration, or to keep the existing index configuration and implement new indexes to improve the performance of a slow-running subset of the queries.

## See Also

<u>Indexes</u>

Parallel Operations Creating Indexes

SQL Server Setup Help

# **View Indexes**

Microsoft® SQL Server<sup>™</sup> 2000 supports defining indexes on views. Views are sometimes called virtual tables because the result set returned by the view has the same general form as a table with columns and rows, and views can be referenced the same way as tables in SQL statements. The result set of a non-indexed view is not stored permanently in the database. Each time a query references the view, SQL Server dynamically merges the logic needed to build the view result set into the logic needed to build the complete query result set from the data in the base tables. The process of building the view results is called materializing the view. For more information, see <u>View Resolution</u>.

For a nonindexed view, the overhead of dynamically building the result set for each query that references a view can be substantial for views that involve complex processing of large numbers of rows. Examples include views that aggregate large amounts of data, or join many rows. If such views are frequently referenced in queries, you can improve performance by creating a unique clustered index on the view. When a unique clustered index is created on a view, the view is executed and the result set is stored in the database in the same way a table with a clustered index is stored. For more information about the structure used to store clustered indexes, see <u>Clustered Indexes</u>.

Another benefit of creating an index on a view is that the optimizer starts using the view index in queries that do not directly name the view in the FROM clause. Existing queries can benefit from the improved efficiency of retrieving data from the indexed view without having to be recoded.

Creating a clustered index on a view stores the result set built at the time the index is created. An indexed view also automatically reflects modifications made to the data in the base tables after the index is created, the same way an index created on a base table does. As modifications are made to the data in the base tables, the data modifications are also reflected in the data stored in the indexed view. The requirement that the view's clustered index be unique improves the efficiency with which SQL Server can find the rows in the index that are affected by any data modification.

You must have set specific SET options before you can create an index on a

view. The query optimizer will not consider the index for any subsequent SQL statements unless the connection executing the statement has the same option settings. For more information, see <u>SET Options That Affect Results</u>.

Indexed views can be more complex to maintain than indexes on base tables. You should create indexes only on views where the improved speed in retrieving results outweighs the increased overhead of making modifications. This usually occurs for views mapped over relatively static data, that process many rows, and are referenced by many queries.

The first index created on a view must be a unique clustered index. After the unique clustered index has been created, you can create additional nonclustered indexes. The naming conventions for indexes on views are the same as for indexes on tables. The only difference is that the table name is replaced with a view name.

All indexes on a view are dropped if the view is dropped. All nonclustered indexes on the view are dropped if the clustered index is dropped. Nonclustered indexes can be dropped individually. Dropping the clustered index on the view removes the stored result set, and the optimizer returns to processing the view like a standard view.

Although only the columns that make up the clustered index key are specified in the CREATE UNIQUE CLUSTERED INDEX statement, the complete result set of the view is stored in the database. As in a clustered index on a base table, the b-tree structure of the clustered index contains only the key columns, but the data rows contain all of the columns in the view result set.

#### See Also

CREATE INDEX
Creating an Indexed View

Using Indexes on Views

# **Maximum Size of Index Keys**

Microsoft® SQL Server<sup>™</sup> 2000 retains the 900-byte limit for the maximum size of an index key but changes the algorithm used by CREATE INDEX to check if the specified index key exceeds the maximum allowable key size of 900 bytes. The new CREATE INDEX algorithm is similar to the row size algorithm used for CREATE TABLE.

Microsoft SQL Server version 7.0 and earlier always used the maximum size of variable columns when checking whether the key specified in a CREATE INDEX statement exceeded 900 bytes, for example:

CREATE TABLE TestTable (PrimaryKey int PRIMARY KEY, VarCharCol1 varchar(500), VarCharCol2 varchar(500) )

-- This statement fails because the maximum sizes

-- of the two columns exceeds 900 bytes:

CREATE INDEX TestIdx ON TestTable(VarCharCol1, VarCharCol2)

In SQL Server 2000, the preceding CREATE INDEX statement succeeds with a warning message, unless one or more rows of data will generate a key whose value exceeds 900 bytes.

The SQL Server 2000 CREATE INDEX statement uses these algorithms:

- If the size of all fixed columns plus the maximum size of all variable columns specified in the CREATE INDEX statement is less than 900 bytes, the CREATE INDEX statement completes successfully with no warnings or errors.
- If the size of all fixed columns plus the maximum size of all variable columns exceeds 900, but the size of all fixed columns plus the minimums of the variable columns is less than 900, the CREATE

INDEX statement succeeds with a warning that a subsequent INSERT or UPDATE statement may fail if it specifies values that generates a key value larger than 900 bytes. The CREATE INDEX statement fails if existing data rows in the table have values that generate a key larger than 900 bytes. A subsequent INSERT or UPDATE statement that specifies data values that generates a key value longer than 900 bytes fails.

• The CREATE INDEX statement fails if the size of all fixed columns plus the minimum size of all variable columns specified in the CREATE INDEX statement exceeds 900 bytes.

This table shows the results of creating indexes where the keys contain only fixed or only variable-length columns.

Index Columns					
Size of the fixed-data column(s)	length	MAX of the SUM of the index key column lengths*	Index created	Message	INSERT or UPDATE run- time error due to oversized index key value
> 900 bytes	None	Not relevant	No	Error	No index present to generate error.
< = 900 bytes	None	Not relevant	Yes	None	No
None	< = 900 bytes	Not relevant	Yes	None	No
None	> 900 bytes	> 900 bytes	No	Error	No index present to generate error.
None	> 900 bytes	< = 900 bytes	Yes	Warning	Only if the sum of current lengths of all index columns is greater than 900 bytes.

\* None of the rows in the table at time the CREATE INDEX statement is executed can have index key values whose total lengths exceed 900 bytes.

This table shows the results of creating indexes where the keys contain a mixture of fixed and variable-length columns.

Index Columns					
Minimum size of variable- length column(s) + Size of the fixed-data column(s)	Maximum size of variable- length column(s) + Size of the fixed-data column(s)	MAX of the SUM of the index key column lengths *	Index created	Message	INSERT or UPDATE run- time error due to oversized index key value
> 900 bytes	Not relevant	Not relevant	No	Error	No index present to generate error.
< = 900 bytes	< = 900 bytes	Not relevant	Yes	None	No.
< = 900 bytes	> 900 bytes	< = 900 bytes	Yes	Warning	Only if the sum of current lengths of all index columns is greater than 900 bytes.
<= 900 bytes	> 900 bytes	> 900 bytes	No	Error	No index present to generate error.

\* None of the rows in the table at time the CREATE INDEX statement is executed can have index key values whose total lengths exceed 900 bytes.

### See Also

**CREATE INDEX** 

# **Property Management**

Microsoft® SQL Server<sup>™</sup> 2000 introduces extended properties that users can define on various objects in a database. These extended properties can be used to store application-specific or site-specific information about the database objects. Because the property is stored in the database, all applications reading the property can evaluate the object in the same way. This helps enforce consistency in how data is treated by all of the programs in the system.

Each extended property has a user-defined name and value. The value of an extended property is a **sql\_variant** that can contain up to 7500 bytes of data. Individual database objects can have multiple extended properties.

Extended properties are managed using three system stored procedures: **sp\_addextendedproperty**, **sp\_updateextendedproperty**, and **sp\_dropextendedproperty**. You can read the value of an existing extended property using the system function **FN\_LISTEXTENDEDPROPERTY**.

There is no convention or standard for defining extended properties. The database designer sets the rules specifying the property names and contents when the database is designed, and then the applications accessing the database have to be coded to follow those rules or conventions.

### See Also

Using Extended Properties on Database Objects

fn\_listextendedproperty

sp\_addextendedproperty

sp\_dropextendedproperty

sp\_updateextendedproperty

# **Full-Text Catalogs and Indexes**

A Microsoft® SQL Server<sup>TM</sup> 2000 full-text index provides efficient support for sophisticated word searches in character string data. The full-text index stores information about significant words and their location within a given column. This information is used to quickly complete full-text queries that search for rows with particular words or combinations of words.

Full-text indexes are contained in full-text catalogs. Each database can contain one or more full-text catalogs. A catalog cannot belong to multiple databases and each catalog can contain full-text indexes for one or more tables. A table can only have one full-text index, so each table with a full-text index belongs to only one full-text catalog.

Full-text catalogs and indexes are not stored in the database to which they belong. The catalogs and indexes are managed separately by the Microsoft Search service.

A full-text index must be defined on a base table; it cannot be defined on a view, system table, or temporary table. A full-text index definition includes:

- A column that uniquely identifies each row in the table (primary or candidate key) and does not allow NULLs.
- One or more character string columns covered by the index.

The full-text index is populated with the key values. The entry for each key has information about the significant words (noise-words or stop-words are stripped out) that are associated with the key, the column they are in, and their location in the column.

Formatted text strings, such as Microsoft® Word<sup>™</sup> document files or HTML files, cannot be stored in character string or Unicode columns because many of the bytes in these files contain data structures that do not form valid characters. Database applications may still have a need to access this data and apply full-text searches to it. Many sites store this type of data in **image** columns, because **image** columns do not require that each byte form a valid character. SQL Server 2000 introduces the ability to perform full-text searches against these types of

data stored in **image** columns. SQL Server 2000 supplies filters that allow it to extract the textual data from Microsoft Office<sup>™</sup> files (.doc, .xls, and .ppt files), text files (.txt files), and HTML files (.htm files). When you design the table, in addition to the **image** column that holds the data, you include a binding column to hold the file extension for the format of data stored in the **image** column. You can create a full-text index that references both the **image** column and the binding column to enable full-text searches on the textual information stored in the **image** column. The SQL Server 2000 full-text search engine uses the file extension information from the binding column to select the proper filter to extract the textual data from the column.

Full-text indexing is the component that implements two Transact-SQL predicates for testing rows against a full-text search condition:

- CONTAINS
- FREETEXT

Transact-SQL also has two functions that return a set of rows that match a full-text search condition:

- CONTAINSTABLE
- FREETEXTTABLE

Internally, SQL Server sends the search condition to the Microsoft Search service. The Microsoft Search service finds all the keys that match the full-text search condition and returns them to SQL Server. SQL Server then uses the list of keys to determine which table rows are to be processed.

### See Also

Full-text IndexesFull-Text Query ArchitectureFull-text Querying SQL Server DataMicrosoft Search Service

## Logins, Users, Roles, and Groups

Logins, users, roles, and groups are the foundation for the security mechanisms of Microsoft® SQL Server<sup>™</sup> 2000. Users that connect to SQL Server must identify themselves using a specific login identifier (ID). Users can then only see the tables and views they are authorized to see, and can only execute the stored procedures and administrative functions they are authorized to execute. This system of security is based on the IDs used to identify users.

### See Also

Managing Security

# Logins

Login identifiers (Ids) are associated with users when they connect to Microsoft® SQL Server<sup>TM</sup> 2000. Login IDs are the accounts that control access to the SQL Server system. A user cannot connect to SQL Server without first specifying a valid login ID. Members of the **sysadmin** fixed server role define login IDs.

**sp\_grantlogin** authorizes a Microsoft Windows® network account (either a group or a user account) to be used as a SQL Server login for connecting to SQL Server using Windows Authentication. **sp\_addlogin** defines a login account for SQL Server connections using SQL Server Authentication.

### See Also

<u>Logins</u> <u>sp\_addlogin</u> <u>sp\_grantlogin</u>

# Users

A user identifier (ID) identifies a user within a database. All permissions and ownership of objects in the database are controlled by the user account. User accounts are specific to a database; the **xyz** user account in the **sales** database is different from the **xyz** user account in the **inventory** database, even though both accounts have the same ID. User IDs are defined by members of the **db\_owner** fixed database role.

A login ID by itself does not give a user permissions to access objects in any databases. A login ID must be associated with a user ID in each database before anyone connecting with that login ID can access objects in the databases. If a login ID has not been explicitly associated with any user ID in a database, it is associated with the **guest** user ID. If a database has no **guest** user account, a login cannot access the database unless it has been associated with a valid user account.

When a user ID is defined, it is associated with a login ID. For example, a member of the **db\_owner** role can associate the Microsoft® Windows® 2000 login **NETDOMAIN\Joe** with user ID **abc** in the **sales** database and user ID **def** in the **employee** database. The default is for the login ID and user ID to be the same.

This example shows giving a Windows 2000 account access to a database and associating the login with a user in the database:

```
USE master
GO
sp_grantlogin 'NETDOMAIN\Sue'
GO
sp_defaultdb @loginame = 'NETDOMAIN\Sue', defdb = 'sales'
GO
USE sales
GO
sp_grantdbaccess 'NETDOMAIN\Sue', 'Sue'
```

### GO

In the **sp\_grantlogin** statement, the Windows 2000 user **NETDOMAIN\Sue** is given access to Microsoft SQL Server<sup>™</sup> 2000. The **sp\_defaultdb** statement makes the **sales** database her default database. The **sp\_grantdbaccess** statement gives the login **NETDOMAIN\Sue** access to the **sales** database and sets her user ID within **sales** to **Sue**.

This example shows defining a SQL Server login, assigning a default database, and associating the login with a user in the database:

```
USE master
GO
sp_addlogin @loginame = 'TempWorker', @password = 'fff', defdb = 's
GO
USE sales
GO
sp_grantdbaccess 'TempWorker'
GO
```

The **sp\_addlogin** statement defines a SQL Server login that will be used by various temporary workers. The statement also specifies the **sales** database as the default database for the login. The **sp\_grantdbaccess** statement grants the **TempWorker** login access to the **sales** database; because no username is specified, it defaults to **TempWorker**.

A user in a database is identified by their user ID, not their login ID. For example, **sa** is a login account mapped to the special user account **dbo** (database owner) in every database. All the security-related Transact-SQL statements use the user ID as the *security\_name* parameter. The administration and understanding of permissions is less confusing if the members of the **sysadmin** fixed server role and the **db\_owner** fixed database role set up the system such that the login ID and user ID of each user are the same, but it is not a requirement.

The **guest** account is a special user account in SQL Server databases. If a user enters a USE *database* statement to access a database in which they are not associated with a user account, they are instead associated with the **guest** user.

## See Also

<u>guest User</u>

<u>sp\_addlogin</u>

<u>sp\_defaultdb</u>

<u>sp\_grantdbaccess</u>

<u>sp\_grantlogin</u>

# Roles

Roles are a powerful tool that allow you to collect users into a single unit against which you can apply permissions. Permissions granted to, denied to, or revoked from a role also apply to any members of the role. You can establish a role that represents a job performed by a class of workers in your organization and grant the appropriate permissions to that role. As workers rotate into the job, you simply add them as a member of the role; as they rotate out of the job, remove them from the role. You do not have to repeatedly grant, deny, and revoke permissions to or from each person as they accept or leave the job. The permissions are applied automatically when the users become members of the role.

Microsoft® Windows NT® and Windows® 2000 groups can be used in much the same way as roles. For more information, see <u>Groups</u>.

It is easy to manage the permissions in a database if you define a set of roles based on job functions and assign each role the permissions that apply to that job. You can then simply move users between roles rather than having to manage the permissions for each individual user. If the function of a job changes, it is easier to simply change the permissions once for the role and have the changes applied automatically to all members of the role.

In Microsoft® SQL Server<sup>™</sup> 2000 and SQL Server version 7.0, users can belong to multiple roles.

The following script shows adding a few logins, users, and roles, and granting permissions to the roles.

```
USE master
GO
sp_grantlogin 'NETDOMAIN\John'
GO
sp_defaultdb 'NETDOMAIN\John', 'courses'
GO
sp_grantlogin 'NETDOMAIN\Sarah'
```

GO

sp\_defaultdb 'NETDOMAIN\Sarah', 'courses' GO

sp\_grantlogin 'NETDOMAIN\Betty'

GO

sp\_defaultdb 'NETDOMAIN\Betty', 'courses' GO

sp\_grantlogin 'NETDOMAIN\Ralph'

GO

sp\_defaultdb 'NETDOMAIN\Ralph', 'courses' GO

sp\_grantlogin 'NETDOMAIN\Diane'

GO

sp\_defaultdb 'NETDOMAIN\Diane', 'courses'

USE courses

GO

sp\_grantdbaccess 'NETDOMAIN\John' GO

sp\_grantdbaccess 'NETDOMAIN\Sarah'

sp\_grantdbaccess 'NETDOMAIN\Betty' GO

sp\_grantdbaccess 'NETDOMAIN\Ralph' GO

sp\_grantdbaccess 'NETDOMAIN\Diane'

GO

sp\_addrole 'Professor'

GO

sp\_addrole 'Student'

GO

sp\_addrolemember 'Professor', 'NETDOMAIN\John' GO

sp\_addrolemember 'Professor', 'NETDOMAIN\Sarah' GO

sp\_addrolemember 'Professor', 'NETDOMAIN\Diane' GO

sp\_addrolemember 'Student', 'NETDOMAIN\Betty' GO

sp\_addrolemember 'Student', 'NETDOMAIN\Ralph' GO

sp\_addrolemember 'Student', 'NETDOMAIN\Diane' GO

GRANT SELECT ON StudentGradeView TO Student GO

GRANT SELECT, UPDATE ON ProfessorGradeView TO Professor GO

This script gives the professors John and Sarah permission to update students' grades, while the students Betty and Ralph can only select their grades. Diane has been added to both roles because she is teaching one class while taking another. The view **ProfessorGradeView** should restrict professors to the rows for students in their classes, while **StudentGradeView** should restrict students to selecting only their own grades.

There are several fixed roles defined in SQL Server 2000 and SQL Server version 7.0 during setup. Users can be added to these roles to pick up the associated administration permissions. These are server-wide roles.

Fixed server role	Description
sysadmin	Can perform any activity in SQL Server.
serveradmin	Can set serverwide configuration options, shut down the server.
setupadmin	Can manage linked servers and startup procedures.
securityadmin	Can manage logins and CREATE DATABASE permissions, also read error logs and change passwords.

processadmin	Can manage processes running in SQL Server.
dbcreator	Can create, alter, and drop databases.
diskadmin	Can manage disk files.
bulkadmin	Can execute BULK INSERT statements.

You can get a list of the fixed server roles from **sp\_helpsrvrole**, and get the specific permissions for each role from **sp\_srvrolepermission**.

Each database has a set of fixed database roles. While roles with the same names exist in each database, the scope of an individual role is only within a specific database. For example, if **Database1** and **Database2** both have user IDs named **UserX**, adding **UserX** in **Database1** to the **db\_owner** fixed database role for **Database1** has no effect on whether **UserX** in **Database2** is a member of the **db\_owner** role for **Database2**.

Fixed database role	Description	
db_owner	Has all permissions in the database.	
db_accessadmin	Can add or remove user IDs.	
db_securityadmin	Can manage all permissions, object ownerships, roles and role memberships.	
db_ddladmin	Can issue ALL DDL, but cannot issue GRANT, REVOKE, or DENY statements.	
db_backupoperator	Can issue DBCC, CHECKPOINT, and BACKUP statements.	
db_datareader	Can select all data from any user table in the database.	
db_datawriter	Can modify any data in any user table in the database.	
db_denydatareader	Cannot select any data from any user table in the database.	
db_denydatawriter	Cannot modify any data in any user table in the database.	

You can get a list of the fixed database roles from **sp\_helpdbfixedrole**, and get the specific permissions for each role from **sp\_dbfixedrolepermission**.

Every user in a database belongs to the **public** database role. If you want everyone in a database to be able to have a specific permission, assign the permission to the **public** role. If a user has not been specifically granted permissions on an object, they use the permissions assigned to **public**.

### See Also

Adding a Member to a Predefined Role sp\_dbfixedrolepermission sp\_helpdbfixedrole sp\_helpsrvrole sp\_srvrolepermission

# Groups

There are no groups in Microsoft® SQL Server<sup>™</sup> 2000 or SQL Server version 7.0. You can, however, manage SQL Server security at the level of an entire Microsoft Windows NT® or Microsoft Windows® 2000 group.

If you use **sp\_grantlogin** and specify the name of a Windows NT or Windows 2000 group, all members of the group can then connect to SQL Server using Windows Authentication.

After the group has been authorized to connect, you can use **sp\_grantdbaccess** to associate the group members with a user identifier (ID) in each database they need to access. You can use two methods:

• Associate the group with a user ID in the database.

In this case, all members of the group will be associated with that user ID when they reference the database.

• Associate an individual user account in the Windows NT or Windows 2000 group with a user ID in the database.

This individual will be associated with the user ID when they reference the database. None of the other individuals in the group will be associated with the user ID. They will be assigned the user ID associated with the group login.

Consider a Windows NT or Windows 2000 group **NETDOMAIN\Managers** with three members: **NETDOMAIN\Sue**, **NETDOMAIN\Fred**, and **NETDOMAIN\Mary**. The following Transact-SQL statements add the Windows NT or Windows 2000 group as both a login and a user in the **sales** database, and then associate **NETDOMAIN\Sue** with a specific user ID:

USE master

GO

- -- Authorize all members of NETDOMAIN\Managers to connect
- -- using Windows Authentication.
- sp\_grantlogin 'NETDOMAIN\Managers'

GO

-- Make sales the default database for all members. sp\_dbdefault 'NETDOMAIN\Managers', 'sales' USE sales

GO

-- Grant all members of the group access to sales

-- No user ID is specified, so SQL Server creates

-- one named 'NETDOMAIN\Managers'

sp\_grantdbaccess 'NETDOMAIN\Managers' GO

-- Grant a specific member of the group access to

-- sales with a specific user.

sp\_grantdbaccess 'NETDOMAIN\Sue', 'Sue'

Permissions can now be granted to either user **NETDOMAIN\Managers** or user **Sue:** 

USE sales GO GRANT SELECT ON SalesTable TO NETDOMAIN\Managers GO GRANT UPDATE ON SalesTable to NETDOMAIN\Sue

The permissions applied to **NETDOMAIN**\**Sue** are the union of the permissions granted, revoked, or denied to both the **NETDOMAIN**\**Managers** or **Sue** users. Any DENY permission overrides any corresponding GRANT permissions.

Unless their Windows NT or Windows 2000 account has been associated with a specific user, members of a group are subject to the permissions assigned to the user associated with the group. If a member of the group creates an object, however, the owner name of the object is their Windows NT or Windows 2000 account name, not the group name. Consider the **NETDOMAIN\Manager** account. If **NETDOMAIN\Fred** connects to the **sales** database, he can see all tables for which **NETDOMAIN\Managers** has been granted SELECT permission. If **NETDOMAIN\Fred** executes the following statement, the table is created as **sales.NETDOMAIN\Fred.TableX**, not

### sales.NETDOMAIN\Managers.TableX:

CREATE TableX (cola INT PRIMARY KEY, colb CHARACTER(200

### See Also

sp\_grantdbaccess

<u>sp\_grantlogin</u>

# **Owners and Permissions**

Every object in Microsoft® SQL Server<sup>™</sup> 2000 is owned by a user. The owner is identified by a database user identifier (ID). When an object is first created, the only user ID that can access the object is the user ID of the owner or creator. For any other user to access the object, the owner must grant permissions to that user. If the owner wants only specific users to access the object, the object, the owner can grant permissions to those specific users.

For tables and views, the owner can grant INSERT, UPDATE, DELETE, SELECT, and REFERENCES permissions, or ALL permissions. A user must have INSERT, UPDATE, DELETE, or SELECT permissions on a table before they can specify it in INSERT, UPDATE, DELETE, or SELECT statements. The REFERENCES permission lets the owner of another table use columns in your table as the target of a REFERENCES FOREIGN KEY constraint from their table. The following example illustrates granting SELECT permissions to a group named **Teachers** and REFERENCES permissions to another development user:

### GRANT SELECT ON MyTable TO Teachers GRANT REFERENCES (PrimaryKeyCol) ON MyTable to DevUser1

The owner of a stored procedure can grant EXECUTE permissions for the stored procedure. If the owner of a base table wants to prevent users from accessing the table directly, they can grant permissions on views or stored procedures referencing the table, but not grant any permissions on the table itself. This is the foundation of the SQL Server mechanisms to ensure that users do not see data they are not authorized to access.

Users can also be granted statement permissions. Some statements, such as CREATE TABLE and CREATE VIEW, can only be executed by certain users (in this case, the **dbo** user). If the **dbo** wants another user to be able to create tables or views, they must grant the permission to execute these statements to that user.

# **Session Context Information**

Microsoft® SQL Server<sup>™</sup> 2000 introduces the ability to programmatically associate up to 128 bytes of binary information with the current session or connection. Session context information enables applications to set binary values that can be referenced in multiple batches, stored procedures, triggers, or user-defined functions operating on the same session, or connection. You can set a session context by using the new SET CONTEXT\_INFO statement, and then you can retrieve the context string from the new **context\_info** column in the **master.dbo.sysprocesses** table.

Session context information differs from Transact-SQL variables, whose scope is limited to the current batch, stored procedure, trigger, or function. Session context information can be used to store information specific to each user or the current state of the application, which can then be used to control the logic in Transact-SQL statements.

The SET CONTEXT\_INFO statement supports:

- A constant, with a maximum of 128 bytes, that is either binary or a data type that can be implicitly converted to binary.
- The name of a **varbinary(128)** or **binary(128)** variable.

SET CONTEXT\_INFO cannot be specified in a user-defined function. You cannot supply a null value to SET CONTEXT\_INFO because the **sysprocesses** table, where the information is stored, does not allow null values.

To get the current session context for the current connection, select the **context\_info** column from the **master.dbo.sysprocesses** row whose SQL Server Process ID (SPID) is equal to the SPID for the connection. The SPID for the current connection is returned by the @@SPID function:

SELECT context\_info FROM master.dbo.sysprocesses WHERE spid = @@SPID The value in the **context\_info** column is initialized to 128 bytes of binary zeros if SET CONTEXT\_INFO has not yet been executed for the current connection. If SET CONTEXT\_INFO has been executed, the **context\_info** column contains the value set by the last execution of SET CONTEXT\_INFO for the current connection. The **context\_info** column is a **varbinary(128)** column.

This is an example of using session context information:

```
-- Set context information at start.
SET CONTEXT INFO 0x1256698456
GO
-- Perform several non-related batches.
sp_who
GO
USE Northwind
GO
SELECT CustomerID
FROM Customers
WHERE City = 'London'
GO
-- Select context information set several batches earlier.
SELECT context info
FROM master.dbo.sysprocesses
WHERE spid = @@spid
GO
```

SET CONTEXT\_INFO does not support referencing expressions other than constants or variable names, such as functions. If you need to set the context information to the result of a function call, you must first place the function call result in a **binary** or **varbinary** variable:

DECLARE @BinVar varbinary(128) SET @BinVar = CAST( REPLICATE( 0x20, 128 ) AS varbinary(128) SET CONTEXT\_INFO @BinVar

See Also

SET CONTEXT\_INFO

<u>sysprocesses</u>

# **Relational Database Components**

The database component of Microsoft® SQL Server<sup>™</sup> 2000 is a Structured Query Language (SQL)–based, scalable, relational database with integrated Extensible Markup Language (XML) support for Internet applications. Each of the following terms describes a fundamental part of the architecture of the SQL Server 2000 database component:

#### Database

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

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

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

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

- Maintaining relationships between data in the database.
- Ensuring that data is stored correctly, and that the rules defining data relationships are not violated.

• Recovering all data to a point of known consistency in case of system failures.

#### **Relational Database**

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

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

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

#### Scalable

SQL Server 2000 supports having a wide range of users access it at the same time. An instance of SQL Server 2000 includes the files that make up a set of databases and a copy of the DBMS software. Applications running on separate computers use a SQL Server 2000 communications component to transmit commands over a network to the SQL Server 2000 instance. When an application connects to an instance of SQL Server 2000, it can reference any of the databases in that instance that the user is authorized to access. The communication component also allows communication between an instance of SQL Server 2000 and an application running on the same computer. You can run multiple instances of SQL Server 2000 on a single computer. SQL Server 2000 is designed to support the traffic of the largest Web sites or enterprise data processing systems. Instances of SQL Server 2000 running on large, multiprocessor servers are capable of supporting connections to thousands of users at the same time. The data in SQL Server tables can be partitioned across multiple servers, so that several multiprocessor computers can cooperate to support the database processing requirements of extremely large systems. These groups of database servers are called federations.

Although SQL Server 2000 is designed to work as the data storage engine for thousands of concurrent users who connect over a network, it is also capable of working as a stand-alone database directly on the same computer as an application. The scalability and ease-of-use features of SQL Server 2000 allow it to work efficiently on a single computer without consuming too many resources or requiring administrative work by the stand-alone user. The same features allow SQL Server 2000 to dynamically acquire the resources required to support thousands of users, while minimizing database administration and tuning. The SQL Server 2000 relational database engine dynamically tunes itself to acquire or free the appropriate computer resources required to support a varying load of users accessing an instance of SQL Server 2000 at any specific time. The SQL Server 2000 relational database engine has features to prevent the logical problems that occur if a user tries to read or modify data currently used by others.

#### **Structured Query Language**

To work with data in a database, you have to use a set of commands and statements (language) defined by the DBMS software. Several different languages can be used with relational databases; the most common is SQL. The American National Standards Institute (ANSI) and the International Standards Organization (ISO) define software standards, including standards for the SQL language. SQL Server 2000 supports the Entry Level of SQL-92, the SQL standard published by ANSI and ISO in 1992. The dialect of SQL supported by Microsoft SQL Server is called Transact-SQL (T-SQL). T-SQL is the primary language used by Microsoft SQL Server applications.

#### **Extensible Markup Language**

XML is the emerging Internet standard for data. XML is a set of tags that can be used to define the structure of a hypertext document. XML documents can be easily processed by the Hypertext Markup Language, which is the most important language for displaying Web pages.

Although most SQL statements return their results in a relational, or tabular, result set, the SQL Server 2000 database component supports a FOR XML clause that returns results as an XML document. SQL Server 2000 also supports XPath queries from Internet and intranet applications. XML documents can be added to SQL Server databases, and the OPENXML clause can be used to expose data from an XML document as a relational result set.

# **Database Applications and Servers**

Microsoft<sup>®</sup> SQL Server<sup>™</sup> 2000 is designed to work effectively as:

- A central database on a server shared by many users who connect to it over a network. The number of users can range from a handful in one workgroup, to thousands of employees in a large enterprise, to hundreds of thousands of Web users.
- A desktop database that services only applications running on the same desktop.

### **Server Database Systems**

Server-based systems are constructed so that a database on a central computer, known as a server, is shared among multiple users. Users access the server through an application:

- In a multitier system, such as Windows® DNA, the client application logic is run in two or more locations:
  - A thin client is run on the user's local computer and is focused on displaying results to the user.
  - The business logic is located in server applications running on a server. Thin clients request functions from the server application, which is itself a multithreaded application capable of working with many concurrent users. The server application is the one that opens connections to the database server. The server application can be running on the same server as the database, or it can connect across the network to a separate server operating as a database server. In complex systems, the business logic may be implemented in several interconnected server applications, or in multiple layers of server applications.

This is a typical scenario for an Internet application. For

example, a multithreaded server application can run on a Microsoft® Internet Information Services (IIS) server and service thousands of thin clients running on the Internet or an intranet. The server application uses a pool of connections to communicate with one or more instances of SQL Server 2000. The instances of SQL Server 2000 can be on the same computer as IIS, or they can be on separate servers in the network.

• In a two-tier client/server system, users run an application on their local computer, known as a <u>client application</u>, that connects over a network to an instance of SQL Server 2000 running on a server computer. The client application runs both business logic and the code to display output to the user, so this is sometimes referred to as a thick client.

### **Advantages of Server Database System**

Having data stored and managed in a central location offers several advantages:

• Each data item is stored in a central location where all users can work with it.

Separate copies of the item are not stored on each client, which eliminates problems with users having to ensure they are all working with the same information. Their system does not need to ensure that all copies of the data are updated with the current values, because there is only one copy in the central location.

• Business and security rules can be defined one time on the server and enforced equally among all users.

Rule enforcement can be done in a database through the use of constraints, stored procedures, and triggers. Rules can also be enforced in a server application, since these applications are also central resources accessed by many thin clients.

• A relational database server optimizes network traffic by returning only the data an application needs.

For example, if an application working with a file server needs to

display a list of the names of sales representatives in Oregon, it must retrieve the entire employee file. If the application is working with a relational database server, it sends this command:

```
SELECT first_name, last_name
FROM employees
WHERE emp_title = 'Sales Representative'
AND emp_state = 'OR'
```

The relational database sends back only the names of the sales representatives in Oregon, not all of the information about all employees.

• Hardware costs can be minimized.

Because the data is not stored on each client, clients do not have to dedicate disk space to storing data. The clients also do not need the processing capacity to manage data locally, and the server does not need to dedicate processing power to displaying data.

The server can be configured to optimize the disk I/O capacities needed to retrieve data, and clients can be configured to optimize the formatting and display of data retrieved from the server.

The server can be stored in a relatively secure location and equipped with devices such as an Uninterruptable Power Supply more economically than fully protecting each client.

• Maintenance tasks such as backing up and restoring data are simplified because they can focus on the central server.

#### Advantages of SQL Server 2000 as a Database Server

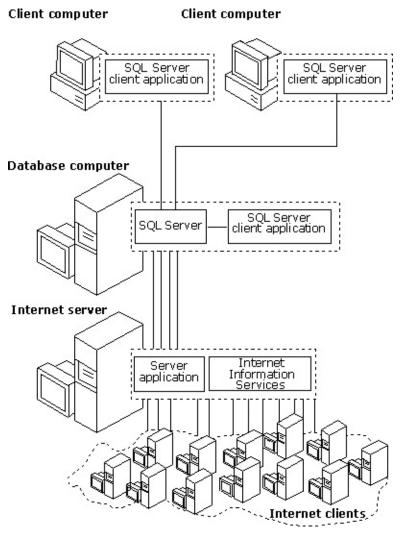
Microsoft SQL Server 2000 is capable of supplying the database services needed by extremely large systems. Large servers may have thousands of users connected to an instance of SQL Server 2000 at the same time. SQL Server 2000 has full protection for these environments, with safeguards that prevent problems, such as having multiple users trying to update the same piece of data at the same time. SQL Server 2000 also allocates the available resources effectively, such as memory, network bandwidth, and disk I/O, among the multiple users.

Extremely large Internet sites can partition their data across multiple servers, spreading the processing load across many computers, and allowing the site to serve thousands of concurrent users.

Multiple instances of SQL Server 2000 can be run on a single computer. For example, an organization that provides database services to many other organizations can run a separate instance of SQL Server 2000 for each customer organization, all on one computer. This isolates the data for each customer organization, while allowing the service organization to reduce costs by only having to administer one server computer.

SQL Server 2000 applications can run on the same computer as SQL Server 2000. The application connects to SQL Server 2000 using Windows Interprocess Communications (IPC) components, such as shared memory, instead of a network. This allows SQL Server 2000 to be used on small systems where an application must store its data locally.

The illustration shows an instance of SQL Server 2000 operating as the database server for both a large Web site and a legacy client/server system.



The largest Web sites and enterprise-level data processing systems often generate more database processing than can be supported on a single computer. In these large systems, the database services are supplied by a group of database servers that form a database services tier. SQL Server 2000 does not support a load-balancing form of clustering for building a database services tier, but it does support a mechanism that can be used to partition data across a group of autonomous servers. Although each server is administered individually, the servers cooperate to spread the database-processing load across the group. A group of autonomous servers that share a workload is called a federation of servers. For more information, see <u>Designing Federated Database Servers</u>.

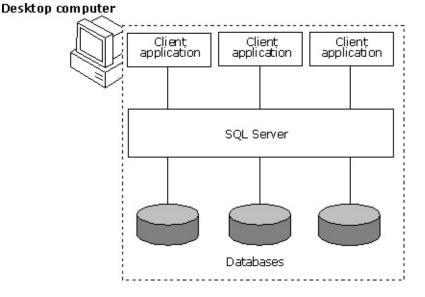
#### **Desktop Database Systems**

Although SQL Server 2000 works effectively as a powerful database server, the

same database engine can also be used in applications that need stand-alone databases stored locally on the client. SQL Server 2000 can configure itself dynamically to run efficiently with the resources available on a client desktop or laptop computer, without the need to dedicate a database administrator to each client. Application vendors can also embed SQL Server 2000 as the data storage component of their applications.

When clients use local SQL Server 2000 databases, applications connect to local instances of the database engine in much the same way they connect across the network to a database engine running on a remote server. The primary difference is that local connections are made through local IPCs such as shared memory, and remote connections must go through a network.

The illustration shows using SQL Server 2000 in a desktop database system.



SQL Server Setup Help

# Logins

To connect to an instance of Microsoft® SQL Server<sup>™</sup> 2000, you typically give an application only two or three pieces of information:

- The network name of the computer on which the SQL Server instance is running.
- The name of the instance (optional, required only if you are connecting to a named instance).
- Your login identifier (ID).

A login ID is the account identifier that controls access to any SQL Server 2000 system. SQL Server 2000 does not complete a connection unless it has first verified that the login ID specified is valid. Verification of the login is called authentication.

One of the properties of a login is the default database. When a login connects to SQL Server, this default database becomes the current database for the connection, unless the connection request specifies that another database be made the current database.

A login ID only enables you to connect to an instance of SQL Server. Permissions within specific databases are controlled by user accounts. The database administrator maps your login account to a user account in any database you are authorized to access. For more information, see <u>Logins, Users,</u> <u>Roles, and Groups</u>.

### **Authenticating Logins**

Instances of SQL Server must verify that the login ID supplied on each connection request is authorized to access the instance. This process is called authentication. SQL Server 2000 uses two types of authentication: Windows Authentication and SQL Server Authentication. Each has a different class of login ID.

#### **Windows Authentication**

A member of the SQL Server 2000 **sysadmin** fixed server role must first specify to SQL Server 2000 all the Microsoft Windows NT® or Microsoft Windows® 2000 accounts or groups that can connect to SQL Server 2000. When using Windows Authentication, you do not have to specify a login ID or password when you connect to SQL Server 2000. Your access to SQL Server 2000 is controlled by your Windows NT or Windows 2000 account or group, which is authenticated when you log on to the Windows operating system on the client.

When you connect, the SQL Server 2000 client software requests a Windows *trusted connection* to SQL Server 2000. Windows does not open a trusted connection unless the client has logged on successfully using a valid Windows account. The properties of a trusted connection include the Windows NT and Windows 2000 group and user accounts of the client that opened the connection. SQL Server 2000 gets the user account information from the trusted connection properties and matches them against the Windows accounts defined as valid SQL Server 2000 logins. If SQL Server 2000 finds a match, it accepts the connection. When you connect to SQL Server 2000 using Windows 2000 Authentication, your identification is your Windows NT or Windows 2000 group or user account.

The Microsoft Windows 98 operating system does not support the server side of the trusted connection API. When SQL Server is running on Windows 98, it does not support Windows Authentication. Users must supply a SQL Server login when they connect. When SQL Server is running on Windows NT or Windows 2000, Windows 95 and Windows 98 clients can connect to it using Windows 2000 Authentication.

#### **SQL Server Authentication**

A member of the **sysadmin** fixed server role first specifies to SQL Server 2000 all the valid SQL Server 2000 login accounts and passwords. These are not related to your Microsoft Windows account or network account. You must supply both the SQL Server 2000 login and password when you connect to SQL Server 2000. You are identified in SQL Server 2000 by your SQL Server 2000 login.

### **SQL Server Authentication Modes**

When SQL Server 2000 is running on Windows NT or Windows 2000, members of the **sysadmin** fixed server role can specify one of two authentication modes:

• Windows Authentication Mode

Only Windows Authentication is allowed. Users cannot specify a SQL Server 2000 login ID. This is the default authentication mode for SQL Server 2000. You cannot specify Windows Authentication Mode for an instance of SQL Server running on Windows 98, because the operating system does not support Windows Authentication.

• Mixed Mode

If users supply a SQL Server 2000 login ID when they log on, they are authenticated using SQL Server Authentication. If they do not supply a SQL Server 2000 login ID, or request Windows Authentication, they are authenticated using Windows Authentication.

These modes are specified during setup or with SQL Server Enterprise Manager.

### **Login Delegation**

If you use Windows Authentication to log on to an instance of SQL Server 2000 running on Windows 2000, and the computer has Kerberos support enabled, SQL Server 2000 can pass your Windows login credentials to other instances of SQL Server. Delegation of your credentials from one instance to another is sometimes called impersonation, typically when both instances of SQL Server are running on the same computer.

For example, if Instance A and Instance B are running on separate computers using Windows 2000, you can connect to Instance A and execute a distributed query that references tables on Instance B. When Instance A connects to Instance B to retrieve the required data, Instance A can use your Windows account credentials for the connection. Instance B has visibility to your specific account, and can validate your individual permissions to access the data requested.

Without delegation, administrators have to specify the login that Instance A uses to connect to Instance B (or any other instance). This login is used regardless of which user executes a distributed query on Instance A, and prevents Instance B

from having any knowledge of the actual user executing the query. The administrators of Instance B cannot define permissions specific to individual users coming in from Instance A, they must define a global set of permissions for the login account used by Instance A. The administrators also cannot audit which specific users perform actions in Instance B. Using delegation with Windows Authentication on Windows 2000 allows administrators greater control over user permissions and gives auditors greater visibility to the actions of individual users.

Connections that use delegation are authenticated using a Kerberos ticket. Each ticket has a timeout period defined by the Windows 2000 security administrator. If a connection remains idle for a long period and the Kerberos ticket times out, all subsequent attempts to execute a distributed query will fail until the user disconnects and reconnects.

#### See Also

Managing Security
Security Account Delegation

SQL Server Setup Help

# **Client Components**

Clients do not access Microsoft® SQL Server<sup>™</sup> 2000 directly; instead, clients use applications written to access the data in SQL Server. These can include utilities that come with SQL Server 2000, third-party applications that access SQL Server 2000, in-house applications developed by programmers at the SQL Server 2000 site, or Web pages. SQL Server 2000 can also be accessed through COM, Microsoft ActiveX®, or Windows® DNA components.

SQL Server 2000 supports two main classes of applications:

- Relational database applications that send Transact-SQL statements to the database engine; results are returned as relational result sets.
- Internet applications that send either Transact-SQL statements or XPath queries to the database engine; results are returned as XML documents.

### **Relational Database APIs**

Relational database applications are written to access SQL Server 2000 through a database application programming interface (API). A database API contains two parts:

• The language statements passed to the database.

The language by relational SQL Server 2000 applications is Transact-SQL. Transact-SQL supports all SQL-92 Entry Level SQL statements and many additional SQL-92 features. It also supports the ODBC extensions to SQL-92 and other extensions specific to Transact-SQL.

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

#### **Native API Support**

Native API support means the API function calls are mapped directly to the

network protocol sent to the server. There is no intermediate translation to another API needed. SQL Server 2000 provides native support for two main classes of database APIs:

• OLE DB

SQL Server 2000 includes a native OLE DB provider. The provider supports applications written using OLE DB, or other APIs that use OLE DB, such as ActiveX Data Objects (ADO). Through the native provider, SQL Server 2000 also supports objects or components using OLE DB, such as ActiveX, ADO, or Windows DNA applications.

• ODBC

SQL Server 2000 includes a native ODBC driver. The driver supports applications or components written using ODBC, or other APIs using ODBC, such as DAO, RDO, and the Microsoft Foundation Classes (MFC) database classes.

An example of nonnative support for an API would be a database that does not have an OLE DB provider, but does have an ODBC driver. An OLE DB application could use the OLE DB provider for ODBC to connect to the database through an ODBC driver. This provider maps the OLE DB API function calls from the application to ODBC function calls it sends to the ODBC driver.

## Additional SQL Server API Support

SQL Server 2000 also supports:

• DB-Library

DB-Library is an API specific to SQL Server 2000 and Microsoft SQL Server. SQL Server 2000 supports DB-Library applications written in C. DB-Library has not been extended beyond the functionality it had in Microsoft SQL Server version 6.5. Existing DB-Library applications developed against earlier versions of Microsoft SQL Server can be run against SQL Server 2000, but many features introduced in SQL Server 2000 and SQL Server version 7.0 are not available to DB-Library applications.

• Embedded SQL

SQL Server 2000 includes a C precompiler for the Embedded SQL API. Embedded SQL applications use the DB-Library DLL to access SQL Server 2000.

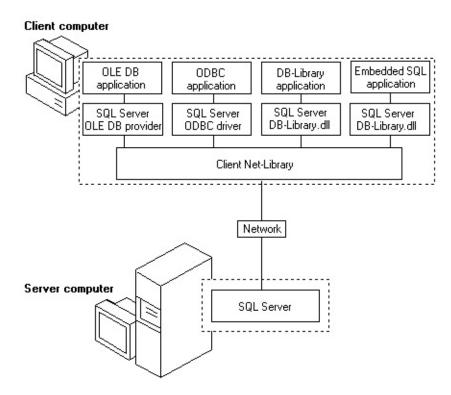
#### XML Access

Internet applications retrieve results in the form of XML documents rather than relational result sets. The applications execute either XPath queries or Transact-SQL statements that use the FOR XML clause to specify that results be returned as XML documents. If you define a virtual root on a Microsoft Internet Information Server (IIS) that points to an instance of SQL Server 2000, IIS applications can use three mechanisms for executing XPath queries or Transact-SQL statements:

- Execute a Uniform Resource Locator (URL) that references the virtual root and contains an XPath query or Transact-SQL statement with FOR XML.
- Use the ADO API to execute an XPath query to Transact-SQL statement with FOR XML.
- Use the OLE DB API to execute an XPath query to Transact-SQL statement with FOR XML.

### **Client Communications**

The Microsoft OLE DB Provider for SQL Server 2000, the SQL Server 2000 ODBC driver, and DB-Library are each implemented as a DLL that communicates to SQL Server 2000 through a component called a client Net-Library.



### See Also

Application Development Architecture

**Overview of Building SQL Server Applications** 

SQL Server Setup Help

# **Communication Components**

Microsoft® SQL Server<sup>™</sup> 2000 supports several methods of communicating between client applications and the server. When the application is on the same computer as an instance of SQL Server 2000, Windows Interprocess Communication (IPC) components, such as local named pipes or shared memory, are used. When the application is on a separate client, a network IPC is used to communicate with SQL Server.

An IPC has two components:

• Application Programming Interface (API)

The API is a definition of the set of functions software uses to send requests to and retrieve results from the IPC.

• Protocol

The protocol defines the format of the information sent between any two components communicating through the IPC. In the case of a network IPC, the protocol defines the format of the packets sent between two computers using the IPC.

Some network APIs can be used over multiple protocols. For example, the Named Pipes API and the Microsoft Win32® RPC API can both be used with several protocols. Other network APIs, such as the Banyan VINES API, can be used with only one protocol.

The SQL Server 2000 client communication components require little or no administration when they connect to SQL Server 2000. Although the actual implementation of the communication components is more complex than in earlier versions of SQL Server, SQL Server 2000 users are shielded from this when connecting to instances of SQL Server 2000. The SQL Server 2000 client software dynamically determines the network address needed to communicate with any instance of SQL Server 2000. All the client software needs is the network name of the computer on which the SQL Server 2000 instance is running, and the name of the instance if connecting to a named instance. There are very few reasons for SQL Server 2000 users to manage the client communications components using the Client Network Utility.

### System Area Networks

SQL Server 2000 Enterprise Edition introduces support for System Area Network (SAN) protocols built using the Virtual Interface Architecture (VIA). A SAN is a high-speed, highly reliable network for interconnecting servers or clusters of servers. A multi-tier, distributed system can generate extremely high levels of network traffic between servers. Gaining high performance in such a system is possible only if message transmissions are fast enough to minimize the time the servers spend processing messages and waiting for replies. Compared to local area networks (LANs) or wide area networks (WANs), SANs support high levels of messaging traffic by lowering CPU loads and message latency. SANs are also more reliable than LANs or WANs, and are implemented in groups or clusters of servers that are located close together, such as in the same computer room.

Compaq®, Intel®, Microsoft, and other companies have defined Virtual Interface Architecture (VIA) as a generic definition of a SAN that allows many possible hardware implementations. The Virtual Interface Architecture allows a VIA provider to implement a flexible, scalable, robust messaging component built at low cost using standard components. VIA SANs can support the intense messaging requirements of large Web servers.

The Virtual Interface Architecture defines both an API and a protocol. The API is referred to as the VIA API, and protocol is referred to as the VIA protocol.

SANs are well suited for these uses with SQL Server 2000:

- The application servers forming the business services tier can use the SAN for high-speed communications with the data services tier. This is done when the application servers and database servers are at the same physical location.
- SQL Server 2000 servers can use the SAN to improve the performance of distributed queries, distributed transactions, and data replication between database servers at the same location. A SAN can improve the distributed queries needed to support the distributed views used to implement federations of computers running SQL Server.

SQL Server 2000 supports the Giganet VIA SAN implementation. Because

SANs are intended to support the high communications bandwidth between servers, SQL Server 2000 only supports the VIA Net-Libraries on the Windows NT® Server, Windows 2000 Data Center, Advanced Server, and Server operating systems. SQL Server Setup Help

# **Client and Server Net-Libraries**

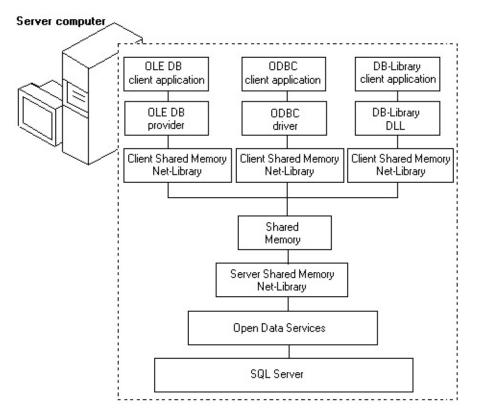
Microsoft® SQL Server<sup>™</sup> 2000 uses components called client Net-Libraries to shield the OLE DB Provider for SQL Server 2000, the SQL Server 2000 ODBC driver, and the DB-Library DLL, from the details of communicating with different Interprocess Communication (IPC) components. Server Net-Libraries perform the same function for the database engine.

The following components manage communications between SQL Server 2000 and its clients in this sequence:

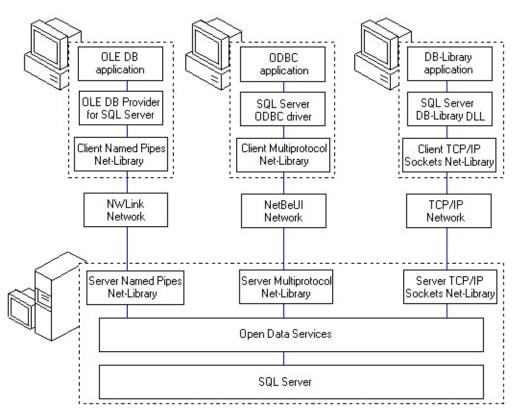
- 1. The client application calls the OLE DB, ODBC, DB-Library, or Embedded SQL API. This causes the OLE DB provider, ODBC driver, or DB-Library DLL to be used for SQL Server communications.
- 2. The OLE DB provider, ODBC driver, or DB-Library DLL calls a client Net-Library. The client Net-Library calls an IPC API.
- 3. The client calls to the IPC API are transmitted to a server Net-Library by the underlying IPC. If it is a local IPC, calls are transmitted using a Windows operating IPC such as shared memory or local named pipes. If it is a network IPC, the network protocol stack on the client uses the network to communicate with the network protocol stack on the server.
- 4. The server Net-Library passes the requests coming from the client to the instance of SQL Server 2000.

Replies from SQL Server 2000 to the client follow the reverse sequence.

This illustration shows the communication path when a SQL Server application runs on the same computer as an instance of SQL Server.



This is a simplified illustration of the communication path when a SQL Server application connects through a LAN or WAN to an instance of SQL Server 2000 on a separate computer. Although the illustration shows the OLE DB Provider for SQL Server 2000, SQL Server 2000 ODBC driver, and DB-Library DLL using specific Net-Libraries, there is nothing that limits these components to these Net-Libraries. The provider, driver, and DB-Library can each use any of the SQL Server Net-Libraries.

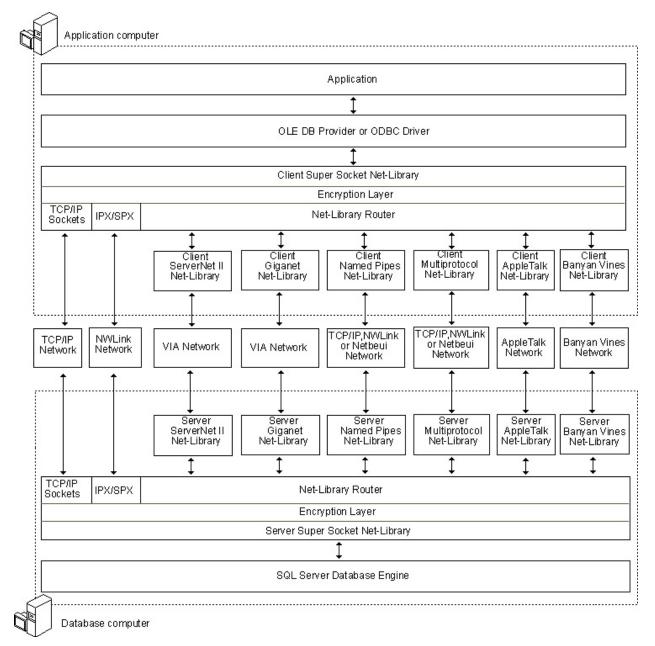


SQL Server 2000 classifies the Net-Libraries as primary or secondary Net-Libraries. The OLE DB Provider for SQL Server 2000, the SQL Server 2000 ODBC driver, the DB-Library DLL, and the database engine communicate directly with only the two primary Net-Libraries:

- By default, local connections between an application and an instance of SQL Server 2000 on the same computer use the Shared Memory primary Net-Library. This path is shown in the illustration above.
- Intercomputer connections communicate through the Super Socket primary Net-Library. The Super Socket Net-Library has two communication paths:
  - If you choose a TCP/IP Sockets connection or an NWLINK IPX/SPX connection, the Super Socket Net-Library directly calls the Windows Socket 2 API for the communication between the application and the instance of SQL Server 2000.
  - If a Named Pipes, Virtual Interface Architecture (VIA) SAN,

Multiprotocol, AppleTalk, or Banyan VINES connection is chosen, a subcomponent of the Super Socket Net-Library, called the Net-Library router, loads the secondary Net-Library for the chosen protocol and routes all Net-Library calls to it.

This illustration shows in more detail the communication paths through the client and server Net-Libraries for network connections between a computer running the SQL Server 2000 client components and an instance of SQL Server 2000.



The server Super Socket Net-Library is implemented as Ssnetlib.dll, and the client Super Socket Net-Library is implemented as Dbnetlib.dll.

This table shows how the Net-Libraries relate to the IPC APIs and protocols used to make connections.

Protocol specified in network utilities		Server Net- Library used	IPC API called by Net- Library	Protocols supporting the IPC API
TCP/IP Sockets	Dbnetlib.dll	Ssnetlib.dll	Windows Socket 2	TCP/IP
Named Pipes	routes to	Ssnetlib.dll routes to Ssnmpn70.dll (Microsoft Windows NT® and Windows® 2000 only)	Windows Named Pipes	File system (local) TCP/IP NetBEUI NWLink
NWLink IPX/SPX	Dbnetlib.dll	Ssnetlib.dll	Windows Socket 2	NWLink
VIA GigaNet SAN	routes to Dbmsgnet.dll (Microsoft Windows NT	Ssnetlib.dll routes to Dbmsgnet.dll (Microsoft Windows NT and Windows 2000 only)	Virtual Interface Architecture (VIA)	Virtual Interface Architecture (VIA)
Multiprotocol	Dbnetlib.dll routes to Dbmsrpcn.dll	Ssnetlib.dll routes to Ssmsrpc.dll (default instance only)	Windows RPC	File system (local) TCP/IP NetBEUI NWLink
AppleTalk	Dbnetlib.dll routes to	Ssnetlib.dll routes to	AppleTalk ADSP	AppleTalk

	Ssmsad70.dll (default instance only)		
routes to Dbmsvinn.dll	routes to	Banyan VINES SPP	Banyan VINES

Instances of SQL Server 2000 running on Microsoft Windows® 98 do not support the server Named Pipes and Banyan VINES Net-Libraries, because the Windows 98 operating system does not support the server part of these APIs. SQL Server 2000 also does not support the server NWLink IPX/SPX Net-Library on Windows 98. SQL Server 2000 does support the client side of these Net-Libraries on Windows 98; therefore, applications running on Windows 98 can use the Net-Libraries to connect to instances of SQL Server on Microsoft Windows NT or Microsoft Windows 2000. Applications running on Windows 95 can also make connections using the client side of these Net-Libraries.

The AppleTalk Net-Library does not run on computers running Windows 95 or Windows 98.

VIA networks are designed to support the high levels of messaging traffic between servers in the same data center, such as in a Web site implemented as one or more Internet Information Services application servers connected to one or more database servers running SQL Server. VIA networks are not used to connect individual workstations. Both the client and server SQL Server VIA Net-Libraries are supported only on Windows NT Server and Advanced Server, and Windows 2000 Server, Advanced Server, and Data Center.

Named instances of SQL Server 2000 support only the Named Pipes, TCP/IP Sockets, NWLink IPX/SPX, and Shared Memory Net-Libraries. Named instances do not support the Multiprotocol, AppleTalk, or Banyan VINES Net-Libraries. To maintain compatibility with earlier versions of SQL Server, default instances support all server Net-Libraries.

Some of the Net-Libraries support only one type of protocol stack. For example,

the AppleTalk Net-Library requires an AppleTalk protocol stack. Other Net-Libraries, such as the Named Pipes and Multiprotocol Net-Libraries support several protocol stacks.

The Microsoft SQL Server Net-Libraries have been tested intensively with the Microsoft protocol stacks and are supported with these stacks. Protocol stacks from other vendors should work, provided that the stacks fully support the APIs used by the Microsoft SQL Server Net-Libraries.

When the Named Pipes or Multiprotocol Net-Libraries are used to connect an application to an instance of SQL Server on the same computer, and the computer does not have a protocol stack, the IPC APIs are implemented by the file system.

SQL Server Setup Help

# **Controlling Net-Libraries and Communications Addresses**

After installing Microsoft® SQL Server<sup>™</sup> 2000, you define the behaviors of the client Net-Libraries by using the Client Network Utility and server Net-Libraries by using the Server Network Utility.

Each instance of SQL Server 2000 can be listening on any combination of the server Net-Libraries at one time. There is one set of server Net-Libraries for each set of database engine executable files. The server Net-Libraries are installed in: C:\Program Files\Microsoft SQL Server\MSSQL\$*n*, where *n* is the number associated with this set of database engine executable files.

All of the server Net-Libraries are installed during the server portion of SQL Server Setup, but some of them may not be active. The person running the Setup program can choose which combination of Net-Libraries is active for the instance being installed. The table shows the default server Net-Libraries that are activated by SQL Server Setup for the Microsoft Windows NT®, Microsoft Windows® 2000, and Microsoft Windows 98 operating systems.

Windows NT and Windows 2000	Windows 98
TCP/IP Sockets	TCP/IP Sockets
Shared Memory	Shared Memory
Named Pipes	

### **Disabling and Enabling Net-Libraries**

After setup, you can disable and enable individual server Net-Libraries for each instance of SQL Server on a database computer using the Server Network Utility. When a server Net-Library is disabled for a specific instance, the database engine for the instance does not load the server Net-Library and does not accept connections using that Net-Library. The server Net-Library remains installed and can be enabled for other instances sharing the same set of executable files. For more information, see <u>SQL Server Network Utility</u>.

There is always one set of the client Net-Library DLLs installed on any computer running SQL Server 2000 client components. The client Net-Library

DLLs are installed in the C:\Windows\System32 or C:\Windows\System directory. All of the client Net-Libraries are installed when you install the SQL Server 2000 client utilities. You can enable and disable the various client Net-Libraries using the Client Network Utility. When a client Net-Library is disabled it remains installed but is not considered for any connections. You can:

- Specify the sequence in which client Net-Libraries are considered for all connections except those that use a server alias.
- Enable or disable specific client Net-Libraries.
- As a compatibility option, define server aliases that define specific Net-Libraries and connection parameters to use when connecting to instances of SQL Server version 7.0 or earlier.

For more information, see <u>Configuring Client Net-Libraries</u>.

#### **Connecting to SQL Server 2000**

For a client to connect to a server running SQL Server 2000, the client must use a client Net-Library that matches one of the server Net-Libraries the server is currently listening on. Also, both the client and server must be running a protocol stack supporting the network API called by the Net-Library being used for the connection. For example, if the client tries using the client Multiprotocol Net-Library, and the server is listening on the server Multiprotocol Net-Library, but the server is running with the TCP/IP protocol while the client computer is running only with the IPX/SPX protocol stack, the client cannot connect to the server. Both the client and the server must be using the same Net-Library and running the same protocol stack.

Each instance of SQL Server on a computer must listen on different network addresses so that applications can connect to specific instances. Default instances of SQL Server 2000 listen on the same default network addresses as earlier versions of SQL Server so that existing client computers can continue to connect to the default instance. The table shows the default network addresses that instances of SQL Server 2000 listen on.

Net-	Default instance network	
Library	address	Named instance network address
TCP/IP	TCP Port 1433	A TCP port is chosen dynamically the fi
Sockets		MSSQL\$ <i>instancename</i> service is started
Named	\\ <i>computername</i> \pipe\sql\query	\\computername\pipe\MSSQL\$instance
Pipes		
NWLink	Port 33854	First available port after 33854 for each
IPX/SPX		
VIA	VIA Port 0:1433	VIA Port 0:1433
Giganet		
SAN		

The VIA server Net-Libraries assign the same default address to both default and named instances. The system administrator must use the Server Network Utility to assign unique port addresses to each instance on a computer.

You can use the SQL Server 2000 Server Network Utility to find out what specific set of network address each instance of SQL Server is listening on for client connections.

When the SQL Server 2000 client Net-Libraries connect to an instance of SQL Server 2000, only the network name of the computer running the instance and the instance name are required. When an application requests a connection to a remote computer, Dbnetlib.dll opens a connection to UDP port 1434 on the computer network name specified in the connection. All computers running an instance of SQL Server 2000 listen on this port. When a client Dbnetlib.dll connects to this port, the server returns a packet listing all the instances running on the server. For each instance, the packet reports the server Net-Libraries and network addresses the instance is listening on. After the Dbnetlib.dll on the application computer receives this packet, it chooses a Net-Library that is enabled on both the application computer and on the instance of SQL Server, and makes a connection to the address listed for that Net-Library in the packet. The connection attempt fails only if:

• The requested instance of SQL Server 2000 is not running.

• None of the Net-Libraries that the instance of SQL Server 2000 is listening on is active on the application computer.

When Dbnetlib.dll compares the network protocols enabled on the application computer against those enabled on the instance of SQL Server 2000, the sequence of the comparison is specified using the Client Network Utility on the application computer. For example, assume an application computer has three client Net-Libraries enabled and specifies that the comparison sequence is TCP/IP Sockets first, NWLink IPX/SPX second, and named pipes third. If the application computer attempts a connection to an instance of SQL Server 2000 that has enabled only the NWLink IPX/SPX, named pipes and Multiprotocol server Net-Libraries, the connection is made using NWLink IPX/SPX. For more information about configuring the comparison sequence, see <u>Configuring Client</u> <u>Net-Libraries</u>.

You cannot assign UDP port 1434 to an application other than SQL Server on computers running instances of SQL Server 2000. Network administrators managing network filters must allow communications on UDP port 1434 to enable SQL Server 2000 connections to pass through the filter.

When running an application on the same computer as a default instance of SQL Server, you can use these names to reference the default instance.

Windows NT and Windows 2000	Windows 98 and Windows 95
Computer name	Computer name
(local)*	(local)*
•	

\*Where "(local)" is the word local in parentheses and "." is a period, or dot. "." is valid only in SQL Server utilities, such as SQL Query Analyzer and **osql**; it cannot be specified in API connection requests.

Do not use either (local) or . to connect to a virtual server implemented using failover clustering.

Using the computer name is recommended. These connections will be made with the Shared Memory Net-Library. DB-Library does not support using (local).

#### **Connecting to Earlier Instances of SQL Server**

When applications using the SQL Server 2000 client components connect to

instances of SQL Server version 7.0 or earlier, the communications between the instance and the application function the same as they did in the earlier versions of SQL Server. Applications using SQL Server version 7.0 or earlier client components to connect to default instances of SQL Server 2000 also communicate as they did in earlier versions of SQL Server. In both of these cases you must administer the network addresses the way they were administered in earlier versions of SQL Server. For more information about configuring a client in earlier versions of SQL Server, see <u>Managing Clients</u>.

SQL Server version 6.5 and earlier supported Windows Authentication (called Integrated Security in those versions) only on the Named Pipes and Multiprotocol Net-Libraries. SQL Server 2000 and SQL Server version 7.0 support Windows Authentication on all Net-Libraries. Existing SQL Server version 6.5 or 7.0 applications that use the default Named Pipes Net-Library can be used to open Windows Authentication connections to instances of SQL Server version 6.5. However, if you upgrade the SQL Server client utilities on the application computer to SQL Server 2000, the default Net-Library changes to TCP/IP, and any attempt to open a Windows Authentication connection to instances of SQL Server version 6.5 fails. To resolve this, you can use the Client Network Utility to put the Named Pipes Net-Library at the top of the Net-Library list, thereby establishing it as the default Net-Library.

#### See Also

Managing Clients

Managing Servers

SQL Server Setup Help

# **Tabular Data Stream Protocol**

Microsoft® SQL Server<sup>™</sup> 2000 uses an application-level protocol called Tabular Data Stream (TDS) for communication between client applications and SQL Server. The TDS packets are encapsulated in the packets built for the protocol stack used by the Net-Libraries. For example, if you are using the TCP/IP Sockets Net-Library, then the TDS packets are encapsulated in the TCP/IP packets of the underlying protocol.

The contents of the packets that send result sets back to the application depends on whether FOR XML is specified in the Transact-SQL statement transmitted to the database engine:

- If FOR XML is not specified, the database engine sends a relational result set back to the application. The TDS packets contain the rows of the result set, with each row comprised of one or more columns, as specified in the select list of the SELECT statement.
- If FOR XML is specified, the database engine streams an XML document back to the application. The XML document is formatting in the TDS packets as if it were a single, long Unicode value, with each packet being approximately 4 KB in size.

You can configure the SQL Server packet size, which is the size of the TDS packets. The size of the TDS packets defaults to 4 KB on most clients (DB-Library applications default to 512 bytes), which testing has shown to be the optimal TDS packet size in almost all scenarios. The size of the TDS packets can be larger than the size of the packets in the underlying protocol. If this is the case, the protocol stack on the sending computer disassembles the TDS packets automatically into units that fit into the protocol packets, and the protocol stack on the client computer reassembles the TDS packets on the receiving computer.

SQL Server Setup Help

# **Net-Library Encryption**

Microsoft® SQL Server<sup>™</sup> 2000 can use the Secure Sockets Layer (SSL) to encrypt all data transmitted between an application computer and a SQL Server instance on a database computer. The SSL encryption is performed within the Super Socket Net-Library (Dbnetlib.dll and Ssnetlib.dll) and applies to all intercomputer protocols supported by SQL Server 2000. When SSL encryption is active, the Super Socket Net-Library performs the SSL encryption before calling:

- The Windows Socket 2 API to transmit TCP/IP Sockets or NWLink IPX/SPX packets.
- The Net-Library router to send a packet to the Named Pipe, Multiprotocol, AppleTalk, or Banyan VINES Net-Libraries.

SSL encryption works only with instances of SQL Server 2000 running on a computer that has been assigned a certificate from a public certification authority. The computer on which the application is running must also have a root CA certificate from the same authority.

The Net-Library encryption is implemented using the Secure Sockets Layer API. The level of encryption, 40-bit or 128-bit, depends on the version of the Microsoft Windows® operating system that is running on the application and database computers.

Enabling encryption slows the performance of the Net-Libraries. Encryption forces these actions in addition to all of the work for an unencrypted connection:

- An extra network round trip is required at connect time.
- All packets sent from the application to the instance of SQL Server must be encrypted by the client Net-Library and decrypted by the server Net-Library.
- All packets sent from the SQL Server instance to the application must

be encrypted by the server Net-Library and decrypted by the client Net-Library.

Shared memory Net-Library communications are inherently secure without the need for encryption. The shared memory Net-Library never participates in intercomputer communications. The area of memory shared between the application process and the database engine process cannot be accessed from any other Windows process.

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 is implemented by calling the Windows RPC encryption API. It does not require the use of certificates. The level of RPC encryption, 40-bit or 128-bit, depends on the version of the Windows operating system that is running on the application and database computers. The Multiprotocol Net-Library is not supported by named instances.

### **Server Components**

In addition to the server Net-Libraries, Microsoft® SQL Server<sup>™</sup> 2000 incorporates these main server components:

- SQL Server database engine (MSSQLServer service)
- SQL Server Agent (SQLServerAgent service)
- Microsoft Search service
- Microsoft Distributed Transaction Coordinator (MS DTC service)

The server components are supported on computers running the Microsoft Windows NT®, Windows® 2000, and Windows 98 operating systems. The server components are not supported on computers running Microsoft Windows 95. When SQL Server is running on Windows NT or Windows 2000, the SQL Server database engine, SQL Server Agent, and MS DTC are implemented as Windows NT or Windows 2000 services. On Windows 98, the server components are not implemented as services because the operating system does not support services. The Microsoft Search service is not available on Windows 95 or Windows 98.

The server components can be stopped and started several ways:

- Windows NT and Windows 2000 can start each service automatically when the operating system is starting.
- Use SQL Server Service Manager to start or stop the service.
- Use SQL Server Enterprise Manager to start or stop the service.
- On Windows NT or Windows 2000, use the **net start** and **net stop** command prompt commands to stop or start each service (except for a

virtual server in a failover cluster).

SQL Server 2000 supports multiple instances of SQL Server on computers running Windows NT or Windows 2000. Each instance has its own copy of the SQL Server service and the SQL Server Agent Service. There are only single copies of the Microsoft Search service or the MS DTC service, whose services are shared among the multiple instances of SQL Server running on the computer.

## **SQL Server Service**

The Microsoft® SQL Server<sup>™</sup> 2000 database engine runs as a service on the Microsoft Windows NT® or Microsoft Windows® 2000 operating systems. It does not run as a service on Microsoft Windows 98 because this operating system does not support services. SQL Server can also run as an executable file on Windows NT and Windows 2000, although it is usually run as a service.

When multiple instances of SQL Server are run on the same computer, each instance has its own SQL Server service. The service name for the default instance is named MSSQLServer, the service name for named instances is MSSQL\$InstanceName. For more information, see <u>Multiple Instances of SQL</u> <u>Server</u>.

The SQL Server service manages all of the files that comprise the databases owned by an instance of SQL Server. It is the component that processes all Transact-SQL statements sent from SQL Server client applications. SQL Server also supports distributed queries that retrieve data from multiple sources, not only SQL Server.

The SQL Server service allocates computer resources effectively between multiple concurrent users. It also enforces business rules defined in stored procedures and triggers, ensures the consistency of the data, and prevents logical problems such as having two people trying to update the same data at the same time.

## **SQL Server Agent Service**

SQL Server Agent supports features allowing the scheduling of periodic activities on Microsoft® SQL Server<sup>™</sup> 2000, or the notification to system administrators of problems that have occurred with the server. The SQL Server Agent components that implement this capability are:

• Jobs

Defined objects consisting of one or more steps to be performed. The steps are Transact-SQL statements that can be executed. Jobs can be scheduled, for example, to execute at specific times or recurring intervals.

• Alerts

Actions to be taken when specific events occur, such as a specific error, errors of certain severities, or a database reaching a defined limit of free space available. The alert can be defined to take such actions as sending an e-mail, paging an operator, or running a job to address the problem.

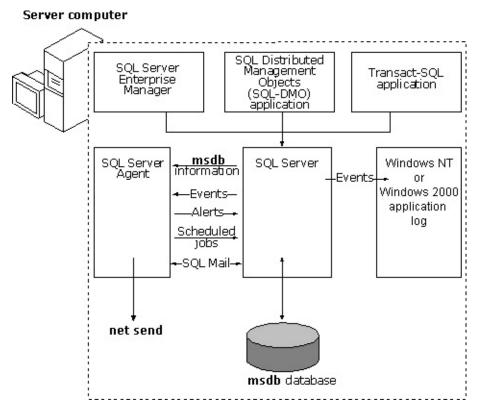
• Operators

People identified through their network account or e-mail identifier (ID) who can address problems with the server. They can be the targets of alerts, either through e-mail, a pager, or a **net send** network command.

The service name of SQLServerAgent applies only to the Agent service associated with a default instance. SQL Server Agent services associated with named instances are named SQLAgent\$InstanceName.

### **Managing Scheduled Operations**

The illustration shows the primary components that are used in the definition and operation of jobs, alerts, and operators.



- Jobs, alerts, and operators are specified using:
  - SQL Server Enterprise Manager.
  - Applications that use SQL Distributed Management Objects (SQL-DMO).
  - Applications that use Transact-SQL and a standard database API.
- The definitions are stored by SQL Server in the **msdb** system database.
- When the SQLServerAgent service is started, it queries the system tables in the **msdb** database to determine what jobs and alerts to enable.
- SQL Server Agent executes jobs at their scheduled time.
- SQL Server passes any events that occur to the SQL Server Agent.

• SQL Server Agent executes any alerts, or sends SQL Mail requests to SQL Server, or sends **net send** commands to Windows.

SQL Server 2000 is more highly automated than SQL Server version 6.5 and earlier, and more efficiently tunes itself to meet processing demands. These features lower the potential for exception conditions that would trigger alerts. Scheduled jobs remain a good feature for implementing recurring tasks such as backup procedures.

#### See Also

Automating Administrative Tasks

## **Microsoft Search Service**

The Microsoft Search service is a full-text indexing and search engine.

The SQL-92 standard defines only basic character-search capabilities:

- For a character value equal to, less than, or greater than a character constant.
- For a character value containing a string pattern.

Using the Microsoft Search service allows Microsoft® SQL Server<sup>™</sup> 2000 and SQL Server version 7.0 to support more sophisticated searches on character string columns.

The Microsoft Search service has two roles:

• Indexing support

Implements the full-text catalogs and indexes defined for a database. Accepts definitions of full-text catalogs, and the tables and columns comprising the indexes in each catalog. Implements requests to populate the full-text indexes.

• Querying support

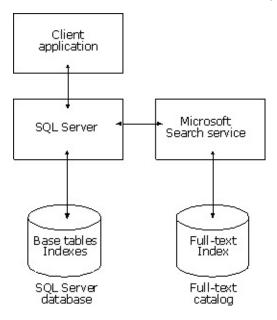
Processes full-text search queries. Determines which entries in the index meet the full-text selection criteria. For each entry that meets the selection criteria, it returns the identity of the row plus a ranking value to the SQL Server service, where this information is used to construct the query result set. The types of queries supported include searching for:

- Words or phrases.
- Words in close proximity to each other.
- Inflectional forms of verbs and nouns.

The full-text engine runs as a service named Microsoft Search on Microsoft Windows NT® or Microsoft Windows® 2000. It is installed when the Full-Text Search feature is selected during custom installation. The Microsoft Search service itself is not installed on Microsoft Windows 95 or Microsoft Windows 98, although Windows 95 and Windows 98 clients can make use of the service when connected to a SQL Server installation running on Windows NT or Windows 2000.

The Microsoft Search service runs in the context of the local system account. During setup, SQL Server adds itself as an administrator of the Microsoft Search service. To ensure this relationship is maintained correctly, all changes to the SQL Server service account information must be made using the **Properties** tab of the **SQL Server Properties** dialog box in SQL Server Enterprise Manager.

The full-text catalogs and indexes are not stored in a SQL Server database. They are stored in separate files managed by the Microsoft Search service. The full-text catalog files are accessible only to the Microsoft Search service and the Windows NT or Windows 2000 system administrator.



### See Also

<u>Full-Text Catalogs and Indexes</u> <u>Full-Text Query Architecture</u>

## **MSSQLServerADHelper Service**

The **MSSQLServerADHelper** service performs two functions:

- It adds and removes the objects used to register instances of Microsoft® SQL Server<sup>™</sup> 2000 relational database engine or Analysis server in the Microsoft Windows® 2000 Active Directory<sup>™</sup>.
- It ensures that the Windows account under which a SQL Server service is running has permissions to update all of the Active Directory objects for the instance, as well as any replication publications and databases for that instance.

The service is dynamically started by an instance of SQL Server or the Analysis Manager when needed. The service is stopped as soon as it has completed its work.

Active Directory objects in a computer container can be created or removed only by programs that have been assigned either domain administration rights or that are running under the **localsystem** Windows account. Few sites run their SQL Server service under either of these types of accounts. A service application that does not perform network administration, such as SQL Server, is rarely granted full domain administration rights. The **localsystem** account cannot be given any privileges on remote computers; therefore, running SQL Server under this account would prevent much of the SQL Server distributed functionality from working. The **MSSQLServerADHelper** service is run under the **localsystem** account so that it can add and remove objects registering SQL Server entities in the Active Directory.

There is only one **MSSQLServerADHelper** service on a computer. The single service handles the Active Directory objects for all instances of the SQL Server relational database engine and all Analysis Manager applications running on the computer.

### **Registering SQL Server Analysis Servers**

Analysis servers are registered from the Analysis Manager, which is a Microsoft

Management Console (MMC) application. When users of Analysis Manager request that an Analysis server be registered in the Active Directory, the application dynamically starts the **MSSQLServerADHelper** service and requests that it create an **MS-SQL-OLAPServer** object in the Active Directory. The helper service is stopped after the object has been completed, and the Analysis Manager finishes filling in the information for the object. For more information, see <u>Using Active Directory with Analysis Services</u>.

### **Registering SQL Server Relational Components**

All management of the registrations of instances of SQL Server, and the databases and replication publications in each instance, are made using system stored procedures on the instance of SQL Server. SQL Server Enterprise Manager calls the system stored procedures when users specify Active Directory actions in the user interface. The procedures used are:

- **sp\_ActiveDirectory\_SCP.** Manages the registration of an instance of the relational database engine.
- **sp\_addpublication**, **sp\_addmergepublication**, **sp\_changepublication**, or **sp\_changemergepublication**. Manage the registration of replication publications.
- **sp\_ActiveDirectory\_Obj.** Manages the registration of a database.

Each of these system stored procedures internally call an internal component that use the Active Directory Services Interface (ADSI) to manage the objects. When an **MS-SQL-SQLServer** object must be added or removed from the Active Directory, or permissions granted, the SQL Server ADSI component calls the **MSSQLServerADHelper** service to perform the task. The SQL Server service uses the SQL Server ADSI component to dynamically start the **MSSQLServerADHelper** service as needed.

The SQL Server service dynamically calls the **MSSQLServerADHelper** service at these times:

• When an **MS-SQL-SQLServer** object must be created in the Active Directory to register an instance of SQL Server, the SQL Server service

calls **MSSQLServerADHelper** to create the object. **MSSQLServerADHelper** creates the object and gives update permissions to the Windows account under which the SQL Server service is running, and then **MSSQLServerADHelper** stops. The SQL Server service now has the permissions needed to maintain the object until it is removed. These permissions include creating **MS-SQL-SQLPublication** and **MS-SQL-SQLDatabase** objects as children of the **MS-SQL-SQLServer** object.

- If an administrator changes the Windows account under which the SQL Server service runs, the SQL Server service detects this the next time it attempts to update any information in objects that existed in the Active Directory before the account change. The SQL Server service automatically starts **MSSQLServerADHelper**. That service reassigns update permissions on the all the objects related to the current instance of SQL Server to the new Windows account.
- When a request is made to delete an **MS-SQL-SQLServer** object, the SQL Server ADSI component calls the **MSSQLServerADHelper** service to delete the object and any children that are still present.

The SQL Server service must be run under a Windows account that has permissions to start the **MSSQLServerADHelper** service. By default, members of the local Power Users and local Administrator's groups have this permission.

# **MS DTC Service**

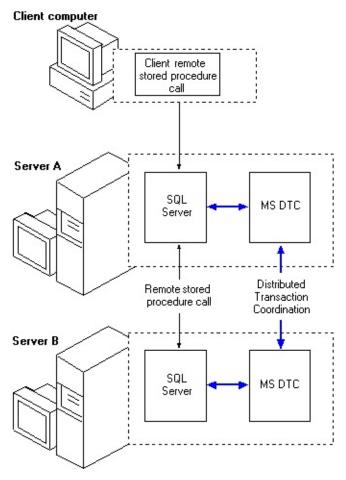
The Microsoft Distributed Transaction Coordinator (MS DTC) is a transaction manager that allows client applications to include several different sources of data in one transaction. MS DTC coordinates committing the distributed transaction across all the servers enlisted in the transaction.

An installation of Microsoft® SQL Server<sup>™</sup> can participate in a distributed transaction by:

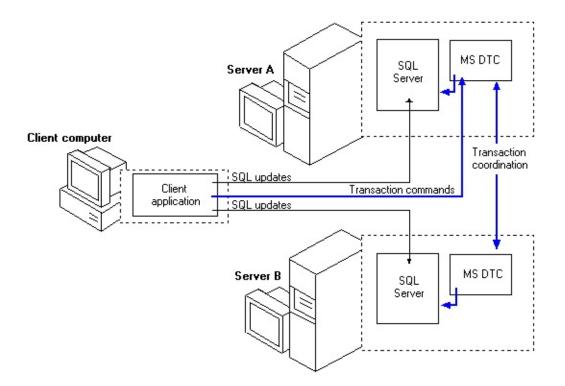
- Calling stored procedures on remote servers running SQL Server.
- Automatically or explicitly promoting the local transaction to a distributed transaction and enlist remote servers in the transaction.
- Making distributed updates that update data on multiple OLE DB data sources.

If these OLE DB data sources support the OLE DB distributed transaction interface, SQL Server can also enlist them in the distributed transaction.

The MS DTC service coordinates the proper completion of the distributed transaction to ensure that either all of the updates on all the servers are made permanent, or, in the case of errors, all erased.



SQL Server applications can also call MS DTC directly to start a distributed transaction explicitly. One or more servers running SQL Server can then be instructed to enlist in the distributed transaction and coordinate the proper completion of the transaction with MS DTC.



### See Also

**Distributed Transactions** 

# **Multiple Instances of SQL Server**

Microsoft® SQL Server<sup>™</sup> 2000 supports multiple instances of the SQL Server database engine running concurrently on the same computer. Each instance of the SQL Server database engine has its own set of system and user databases that are not shared between instances. Applications can connect to each SQL Server database engine instance on a computer in much the same way they connect to SQL Server database engines running on different computers.

There are two types of instances of SQL Server:

#### **Default Instances**

The default instance of the SQL Server 2000 database engine operates the same way as the database engines in earlier versions of SQL Server. The default instance is identified solely by the name of the computer on which the instance is running, it does not have a separate instance name. When applications specify only the computer name in their requests to connect to SQL Server, the SQL Server client components attempt to connect to the default instance of the database engine on that computer. This preserves compatibility with existing SQL Server applications.

There can only be one default instance on any computer, the default instance can be any version of SQL Server.

#### Named Instances

All instances of the database engine other than the default instance are identified by an instance name specified during installation of the instance. Applications must provide both the computer name and the instance name of any named instance to which they are attempting to connect. The computer name and instance name are specified in the format *computer\_name\instance\_name*.

There can be multiple named instances running on a computer, but only the SQL Server 2000 database engine can operate as a named instance. The database engines from earlier versions of SQL Server cannot operate as a named instance.

Instances apply primarily to the database engine and its supporting components,

not to the client tools. When you install multiple instances, each instance gets a unique set of:

- System and user databases.
- The SQL Server and SQL Server Agent services. For default instances, the names of the services remain MSSQLServer and SQLServerAgent. For named instances, the names of the services are changed to MSSQL\$*instancename* and SQLAgent\$*instancename*, allowing them to be started and stopped independently of the other instances on the server. The database engines for the different instances are started and stopped using the associated SQL Server service. The SQL Server Agent services manage scheduled events for the associated instances of the database engine.
- The registry keys associated with the database engine and the SQL Server and SQL Server Agent services.
- Network connection addresses so that applications can connect to specific instances.

### **Shared Components**

The following components are shared between all of the instances running on the same computer:

• There is only one SQL Server 2000 program group (Microsoft SQL Server) on the computer, and only one copy of the utility represented by each icon in the program group. There is only one copy of SQL Server Books Online.

The versions of the utilities in the program group are from the first version of SQL Server 2000 installed on the computer. For example, if you install the French version of SQL Server 2000 as a default instance and then the U.S. English version of SQL Server 2000 as a named instance, there is one SQL Server 2000 program group. All of the utility icons and the SQL Server Books Online icon in the program group start the French versions of the tools.

All of the SQL Server 2000 utilities work with multiple instances. You can start and stop each of the instances from a single copy of the SQL Server 2000 Service Manager. You can use a single copy of the SQL Server 2000 SQL Server Enterprise Manager to control objects in all instances on the computer, and use a single copy of the SQL Server 2000 Server Network Manager to manage the network addresses with which all of the instances on the computer communicate.

- There is only one copy of the MSSearchService that manages full-text searches against all of the instances of SQL Server on the computer.
- There is only one copy each of the English Query and Microsoft SQL Server 2000 Analysis Services servers.
- The registry keys associated with the client software are not duplicated between instances.
- There is only one copy of the SQL Server development libraries (include and .lib files) and sample applications.

#### **Default Instances**

Configurations that can operate as a default instance include:

- A default instance of SQL Server 2000.
- An installation of SQL Server version 7.0 operates as a default instance.
- An installation of SQL Server version 6.5 operates as a default instance.
- A default instance of SQL Server 2000 that can be version switched with an installation of SQL Server version 6.5 using the SQL Server 2000 **vswitch** utility.

• An installation of SQL Server version 7.0 that can be version switched with an installation of SQL Server version 6.5 using the SQL Server version 7.0 **vswitch** utility.

**Note** You must apply SQL Server 6.5 Service Pack 5 to any instance of SQL Server 6.5 before installing instances of SQL Server 2000 on the same computer.

### Switching Between Versions of SQL Server

You cannot version switch between an installation of SQL Server version 7.0 and a default instance of SQL Server 2000.

You can have any number of named instances of SQL Server 2000 in addition to the default instance. You are not required to run a default instance on a computer before you can run named instances. You can run named instances on a computer that has no default instance. SQL Server version 6.5 and SQL Server 7.0 cannot operate as named instances, only as default instances.

Microsoft does not support more than 16 instances on a single computer or failover cluster.

If you run SQL Server version 6.5 as a default instance and run one or more named instances of SQL Server 2000 on a single computer, the computer has two SQL Server program groups instead of one SQL Server program group:

- A SQL Server 2000 program group executes the SQL Server 2000 tools.
- A SQL Server version 6.5 program group runs the SQL Server 6.5 tools.

If you are running SQL Server version 7.0 with SQL Server 2000, the icons in the SQL Server 7.0 program group will execute the SQL Server 2000 tools.

**Note** You must apply SQL Server 6.5 Service Pack 5 to any instance of SQL Server 6.5 before installing instances of SQL Server 2000 on the same computer.

### **Multiple Instances of SQL Server on a Failover Cluster**

You can run only one instance of SQL Server on each virtual server of a SQL Server failover cluster, although you can install up to 16 virtual servers on a failover cluster. The instance can be either a default instance or a named instance. The virtual server looks like a single computer to applications connecting to that instance of SQL Server. When applications connect to the virtual server, they use the same convention as when connecting to any instance of SQL Server; they specify the virtual server name of the cluster and the optional instance name (only needed for named instances): *virtualservername\instancename*. For more information about clustering, see Failover Clustering Architecture.

## **Communicating with Multiple Instances**

Each instance of Microsoft® SQL Server<sup>™</sup> 2000 listens on a unique set of network address so that applications can connect to different instances. SQL Server 2000 clients do not have to be configured to connect to an instance of SQL Server 2000. The SQL Server 2000 client components query a computer running instances of SQL Server 2000 to determine the Net-Libraries and network addresses for each instance. The client components then transparently choose a supported Net-Library and address for the connection without having to be configured on the client. The only information the application must supply is the computer name and instance name. For more information, see <u>Controlling Net-Libraries and Communications Addresses</u>.

A default instance of SQL Server 2000 listens on the same network addresses as earlier versions of SQL Server; therefore, applications using the client connectivity components of SQL Server version 7.0 or earlier can continue to connect to the default instance with no change. Named instances listen on alternative network addresses, and client computers using the client connectivity components of SQL Server version 7.0 or earlier must be set up to connect to the alternative addresses.

# **Using Multiple Instances**

Although running multiple instances of Microsoft® SQL Server<sup>™</sup> 2000 on a single computer expands the capabilities of SQL Server, the recommended configuration for most production databases servers is to use a single instance of SQL Server with multiple databases.

Using a single instance of SQL Server on a production server offers these benefits:

- Only one instance needs to be administered.
- There is no duplication of components or processing overhead, such as having to run multiple database engines on the same computer. This means that the overall performance of a server with a single instance may be higher than a server running multiple instances.
- A single instance of SQL Server 2000 is capable of handling the processing growth requirements of the largest Web sites and enterprise data-processing systems, especially when it is part of a federation of database servers. For more information, see <u>Federated SQL Server 2000</u> <u>Database Servers</u>.

Running multiple instances of SQL Server on a single computer is best:

- When you must support different systems that have to be securely isolated from each other, such as when a service bureau has a large server and must create a separate instance of SQL Server for each customer.
- When you need to support multiple test and development databases, and the most economical configuration is to run these as separate instances of SQL Server on a single large server.

• When you need to run multiple applications on a desktop, and each application installs a separate instance of SQL Server 2000 Desktop Engine.

# **Working with Multiple Instances**

Although multiple instances of Microsoft® SQL Server<sup>™</sup> 2000 can run on a single computer, there is no direct connection between instances. Each instance operates in many ways as if it is on a separate server. An application connected to one instance cannot access objects in databases created in another instance, except through distributed queries. Databases and database files cannot be shared between instances.

Named instances of SQL Server 2000 database engines have almost the same behaviors as default instances. The main difference is that you must supply both the computer name and instance name to identify a named instance. When you specify only *computername*, you work with the default instance. When you specify *computername*\instancename you work with the named instance.

• Service Manager.

When you specify only *computername* in Service Manager, you can stop and start the default instance. When you specify *computername\instancename* you can stop and start the named instance. When a specific instance is started, any database created in that instance is available to any application that connects to the instance using an authorization ID that has permissions to access the database.

• SQL Server Enterprise Manager.

Using SQL Server Enterprise Manager you can register each instance for which you have permissions. After an instance is registered, you can create, edit, and drop objects in the databases associated with that instance, subject to the permissions granted to you. You can also create, edit, and drop Data Transformation Services, Replication, and SQL Server Agent objects for that instance.

• Applications.

In an application, when you specify *computername* as the server name parameter in a connection request, you are connected to the default instance on the computer. You can access any databases in the default instance that you have permissions to access. If you specify *computername\instancename* as the server name parameter, you are connected to the named instance. You can access any databases in that named instance that you have permissions to access. When you are connected to a specific instance, objects in databases in other instances can be accessed only through distributed queries, just as objects in databases on other servers can be accessed only through distributed queries. Applications specify the instance name in different ways:

- ADO applications specify "Server=*computername*\*instancename*" in the provider string. For more information, see <u>Connecting to Multiple Instances of</u> <u>SQL Server</u>.
- OLE DB applications specify
   "Server=computername\instancename" in the provider string.
   They can alternatively set DBPROP\_INIT\_DATASOURCE to
   computername\\instancename (the backslash must be escaped
   with a second backslash). For more information, see
   <u>Establishing a Connection to a Data Source</u>.
- ODBC applications specify

"Server=*computername*\*instancename*" in the connection string specified on **SQLDriverConnect**. They can alternatively specify *computername*\*instancename* for the *ServerName* parameter on **SQLConnect**, or connect through a data source that has *computername*\*instancename* specified for the server name. For more information, see Support for <u>SQLDriverConnect</u> and <u>SQLConfigDataSource</u>.

- SQL DMO applications can manage instances of SQL Server 2000 using the **SQLServer2** object. For more information, see <u>SQLServer2 Object</u>.
- DB-Library and Embedded SQL for C do not support multiple

instances.

• Distributed queries and linked servers.

Distributed queries and linked server definitions use *computername*\*instancename* to identify named instances and *computername* to identify default instances. For more information, see <u>Distributed Queries on Multiple Instances of SQL Server</u>.

• Command prompt utilities.

When you use the command prompt utilities, you can use the Server switch to specify an instance by using *computername*\instancename, for example:

osql -E -Scomputer1\instance1 sqlservr /Sinstance1

The **isql** utility does not support named instances.

• SQL Server 2000 client components.

Applications using SQL Server 2000 client components can enumerate the instances available for connections:

- The OLE DB Provider for SQL Server 2000 returns instance names using **ISourcesRowset::GetSourcesRowset**. The names of named instances are returned as the data source name in the format *computername\instancename*, where *computername* can be either the name of a single computer or the virtual server name of a failover cluster. The names of default instances are returned as the data source name in the format *computername*, with no instance name.
- The SQL Server 2000 ODBC driver supports extensions to **SQLBrowseConnect** and **SQLSetConnectAttr** that allow applications to enumerate instances on a server. ODBC applications can also determine whether the *computername* is the name of a single computer or a virtual server name for a failover cluster. For more information, see

#### SQLBrowseConnect.

- SQL-DMO applications can enumerate instances using the **SQLServer2** object. The **SQLServer2** object also presents information such as the names of the SQL Server and SQL Server Agent services for the instance, or whether the instance is running on a single computer or a failover cluster. For more information, see <u>SQLServer2 Object</u>.
- DB-Library and Embedded SQL for C do not support named instances.

### **Identifying Instances**

Performance Monitor counters, Profiler events, and Windows events in the Event Viewer Application Log all identify the instance of SQL Server with which they are associated.

The string returned by the @@SERVERNAME function identifies the name of the instance in the form *servername*\*instancename* if you are connected to a named instance. If connected to a default instance @@SERVERNAME returns only *servername*. For more information, see <u>@@SERVERNAME</u>.

The SERVERPROPERTY function INSTANCENAME property reports the instance name of the instance to which you are connected. INSTANCENAME returns NULL if connected to a default instance. In addition, the SERVERNAME property returns the same format string returned by @@SERVERNAME and will have the format *servername\instancename* when connected to a named instance. For more information, see <u>SERVERPROPERTY</u>.

Although the strings reported by @@SERVERNAME and SERVERNAME use the same format, the information they report can be different, for example:

• The string returned by @@SERVERNAME is affected by the actions of **sp\_addserver** and **sp\_dropserver**, and the string reported by SERVERNAME is not.

• SERVERNAME automatically reports changes in the network name of the computer, and @@SERVERNAME does not, unless **sp\_dropserver** and **sp\_addserver** are used to change the name it reports.

### **Federated SQL Server 2000 Servers**

Microsoft® SQL Server<sup>™</sup> 2000 databases can be spread across a group of autonomous database servers capable of supporting the processing growth requirements of the largest Web sites and enterprise data-processing systems built with Microsoft Windows® DNA.

Windows DNA divides the processing units of a data processing system into logical tiers:

• User services tier

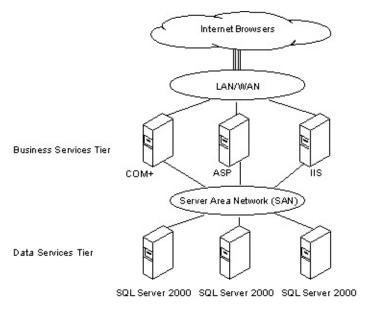
Presents the interface seen by the users, and typically calls the second tier for business logic processing.

• Business services tier

Contains the business logic that controls the operation of the Web site, and uses the persistent data storage provided by the third tier.

• Data services tier

Stores the persistent data required to run the Web site.



Scaling refers to the process of adding resources to a tier so that it can handle increased workloads. Scaling can be done in one of two ways:

• Scale up

Increases the processing power of a server by using a more powerful computer.

• Scale out

Increases the processing power of a system designed in a modular fashion, such as becoming a cluster of computers, by adding one or more additional computers, or nodes, to the system.

The growth requirements of the largest Web sites generate processing loads that exceed the capacity of large individual servers. In these cases, scaling out may be the best option for increasing the processing capacity of the system.

Microsoft Windows 2000 COM+ components are designed to be used in clusters of Windows 2000 application servers to form a clustered business services tier. Each server has identical sets of COM+ components, and Windows 2000 balances the cluster processing load by sending new requests to the server that has the least processing load. This forms an easily administered cluster that can quickly scale out by simply adding a new server.

SQL Server 2000 does not support this type of clustering. However, SQL Server 2000 does support updatable distributed partitioned views used to transparently partition data horizontally across a group of servers. Although these servers cooperate in managing the partitioned data, they operate autonomously. Each server is managed independently, has separate operational rules, and can support independent processes and data. A group of autonomous servers that cooperate to process a workload is known as a federation. Although SQL Server 2000 delivers very impressive performance when scaled up on servers with eight or more processors, it can support huge processing loads when partitioned across a federation. A federation of servers running SQL Server 2000 is capable of supporting the growth requirements of any Web site, or of the largest enterprise systems.

## **Partitioning Data**

The first step in building a set of federated database servers is to horizontally partition the data in a set of tables across multiple servers. Horizontally partitioning a table refers to dividing a table into multiple smaller tables, called member tables. Each member table has the same format as the original table, but only part of the rows. Each table is placed on a separate resource (files or servers) to spread the processing load across the resources. For example, a company assigns customer identifiers (IDs) from 1 through 9999999. The **Customers** table may be partitioned into three member tables, with each member table having an equal customer ID range.

If used without views, horizontal partitioning would require applications to have logic to determine which member tables have the data requested by the user and dynamically build SQL statements referencing the tables. The application would require complex queries joining the member tables. Changing the member tables would also involve recoding the application. Views solve the problem by making the member tables look like one table. The SQL UNION operator combines result sets with identical formats into one. Because all the member tables have the same format, the result of SELECT \* statements for each table have the same format, and can be combined using the UNION clause to form a single result set that operates similarly to the original table. For example, the **Customers** table has been partitioned across three servers (**Server1**, **Server2**, and **Server3**). The distributed partitioned view defined on **Server1** is:

```
CREATE VIEW Customers
AS
SELECT * FROM Customers_33
UNION ALL
SELECT * FROM Server2.CustomerDB.dbo.Customers_66
UNION ALL
SELECT * FROM Server3.CustomerDB.dbo.Customers_99
```

This view makes the actual location of the data transparent to an application. When a SQL statement is executed on **Server1** that references the **Customers** 

partitioned view, the application has no visibility to where the data is located. If some of the rows required to complete the SQL statement reside on **Server2** or **Server3**, the instance of SQL Server on **Server1** automatically generates a distributed query that pulls in the required rows from the other servers. This transparency allows database administrators to repartition tables without recoding applications. If the **Customers** view is updatable, the behavior of the view is the same as a table named **Customers**.

Local partitioned views reference member tables on one server. Distributed partitioned views reference member tables on multiple servers. A server containing a member table is called a member server, and a database containing a member table is called a member database. Each member server contains one member table and a distributed partitioned view. An application that references the partitioned view on any of the servers gets the same results as if a complete copy of the original table were present on each server.

Microsoft SQL Server 2000 and Microsoft SQL Server version 7.0 support partitioned views; however, SQL Server 2000 introduces key features that allow the views to scale out and form federations of database servers:

- SQL Server 2000 partitioned views are updatable. This is crucial for distributing data so that the location of the data is transparent to the application. Updatable views support the full behavior of the original table; nonupdatable views are like read-only copies.
- The SQL Server 2000 query optimizer supports new optimizations that minimize the amount of distributed data that has to be transferred. The distributed execution plans generated by SQL Server 2000 result in good performance for a larger set of queries than the plans generated by SQL Server version 7.0.

SQL Server 2000 partitioned views are best suited for the types of SQL statements generated by Web sites and online transaction processing (OLTP) systems.

### **Partitioning a Database**

To build an effective federation of database servers:

- Create multiple databases, each on a different member server running an instance of SQL Server 2000.
- Partition the individual tables in the original database so that most related data is placed together on a member server. This may require different methods of distributing the data in the various tables across all the member databases; partitioning some tables; making complete copies of other tables in each member database; and leaving some tables intact on the original server.
- Devise data routing rules that can be incorporated in the business services tier, so that applications can send each SQL statement to the member server that stores most of the data required by the statement.

The most important goal is to minimize distributed processing in such a system. You must be able to collocate related data on the same member server, and then route each SQL statement to a member server that contains most, if not all, of the data required to process the statement. For example, you may find that all the sales, customer, sales personnel, and inventory tables in a database can be partitioned by sales region, and that most SQL statements only reference data in a single region. You can then create member servers where each server has the horizontally partitioned data for one or more regions. If applications can identify the region currently referenced in the user's input, the application can submit any generated SQL statement to the member server containing the data for that region. The only SQL statements that will generate distributed queries are those that reference data from multiple regions.

## **Failover Clustering Architecture**

Microsoft® SQL Server<sup>™</sup> 2000 failover clustering increases server availability by allowing a system to automatically switch the processing for an instance of SQL Server from a failed server to a working server. For example, an instance of SQL Server can quickly restore database services to a Web site or enterprise network even if the server running the instance fails. SQL Server 2000 implements failover clustering based on the failover clustering features of the Microsoft Clustering Service (MSCS) in Windows NT® 4.0 and Windows® 2000.

The type of MSCS failover cluster used by SQL Server 2000 consists of multiple server computers (two on Windows NT 4.0, up to four on Windows 2000 Datacenter Server) that share a common set of cluster resources, such as disk drives. Each server in the cluster is called a node. Each server, or node, is connected to the network, and each node can communicate with each other node. Each node runs the same version of MSCS.

The shared resources in the failover cluster are collected into cluster groups. For example, if a failover cluster has four clustered disk drives, two of the drives can be collected in one cluster group and the other two in a second cluster group. Each cluster group is owned by one of the nodes in the failover cluster, although the ownership can be transferred between nodes.

Applications can be installed on the nodes in the failover cluster. These applications are typically server applications or distributed COM objects that users access through network connections. The application executables and other resources are typically stored in one or more of the cluster groups owned by the node. Each node can have multiple applications installed on it.

The failover cluster nodes periodically send each other network messages called heartbeat messages. If the MSCS software detects the loss of a heartbeat signal from one of the nodes in the cluster, it treats the server as a failed server. MSCS then automatically transfers the cluster groups and application resources of that node to the other nodes in the network. The cluster administrator specifies the alternate nodes to which cluster groups are transferred when any given node fails. The other nodes then continue processing user network requests for the applications transferred from the failed server. For more information about MSCS, see the Windows NT Server, Windows 2000 Server, Windows 2000 Advanced Server, or Windows 2000 Datacenter documentation.

### **SQL Server 2000 Failover Clusters**

You can install up to 16 instances of Microsoft® SQL Server<sup>™</sup> 2000 in a Microsoft Clustering Service (MSCS) failover cluster.

You install an instance of SQL Server 2000 by running SQL Server Setup on one of the nodes of the cluster. The Setup program installs the instance on the nodes of the failover cluster that you specify during setup. The SQL Server 2000 executable files are installed on the local disk drives of each node in the failover cluster. This means that each node must have a local hard drive that is assigned the same drive letter as on all the other nodes, and that drive letter must be in the path of the location you specify for the SQL Server executable files during setup. For example, if you specify C:\Program Files\Microsoft SQL Server as the location in which to install the SQL Server executables, each node in the cluster must have drive letter C mapped to a local drive. The registry information for the instance is also stored in the registry of each node in the failover cluster.

An MSCS cluster group is a collection of clustered resources, such as clustered disk drives, which are owned by one of the failover cluster nodes. The ownership of the group can be transferred from one node to another, but each group can only be owned by one node at a time. The database files for an instance of SQL Server 2000 are placed in a single MSCS cluster group owned by the node on which you install the instance. If a node running an instance of SQL Server fails, MSCS switches the cluster group containing the data files for that instance to another node. Since the new node already has the executable files and registry information for that instance of SQL Server on its local disk drive, it can start up the instance of SQL Server and start accepting connection requests for that instance.

Because the executable files and registry information for each instance of SQL Server 2000 is stored in each node, the SQL Server 2000 limit of 16 instances per computer also applies to each failover cluster. Each instance in the failover cluster must either have a unique instance name or be a default instance. There can only be one default instance per failover cluster.

The MSCS cluster group that holds the database files for an instance is associated with a SQL Server virtual server name during SQL Server setup.

There can only be one instance per virtual server, which also means that there can only be one instance associated with any cluster group.

When an application attempts to connect to an instance of SQL Server 2000 running on a failover cluster, the application must specify both the virtual server name and the instance name. The application does not have to specify an instance name only if the instance associated with the virtual server is a default instance that does not have a name.

For example:

- A Windows cluster administrator creates a failover cluster with two nodes: **NodeA** and **NodeB**. Each node maps the drive letter C to a local hard drive.
- There is one shared disk in the cluster. The cluster administrator creates **ClusterGroupA** to hold the drive, and assigns it to **NodeA**.
- The SQL Server system administrator runs the Setup program to install a default instance of SQL Server on **NodeA**. During setup, the administrator specifies a SQL Server virtual server name of **VirtualServerX**, and specifies that the database files be placed on the drive in **ClusterGroupA**. Setup installs the SQL Server executable files on the local drives of both **NodeA** and **NodeB**, and places the database files in **ClusterGroupA**.
- Applications attempting to connect to the default instance only need to specify the virtual server name **VirtualServerA**. The default instance normally runs on **NodeA**. Should **NodeA** fail, however, the MSCS clustering will transfer ownership of **ClusterGroupA** to **NodeB** and will restart the default instance on **NodeB**. Applications will still connect to the default instance by specifying the virtual server name **VirtualServerX**.

### See Also

Failover Clustering

Installing a Virtual Server Configuration

## **Active Directory Integration**

The Microsoft® Windows® 2000 Active Directory<sup>™</sup> operates as a secure central resource for storing information about the users, devices, and services available on a Windows 2000 network. Microsoft SQL Server<sup>™</sup> 2000 supports registering instances of the SQL Server relational engine, databases, replication publications, and Analysis servers in the Active Directory. The SQL Server tools also provide a dialog box that supports browsing for replication publications registered in the Active Directory.

### SQL Server Objects in the Active Directory Hierarchy

The Active Directory uses a hierarchy to represent the relationships between network entities such as users, services, and devices (such as computers, scanners, or printers). The hierarchy starts from a single root node at the top and branches down to leaf nodes representing individual entities in the network. The intermediate nodes in the hierarchy are containers that hold references to multiple entities. For example, several Windows users can be collected into a group for administrative purposes. Each node is implemented as an Active Directory object that represents the specific entity for that node.

When you register an instance of the SQL Server relational database engine in the Active Directory, an **MS-SQL-SQLServer** object is added as a Service Connection Point (SCP) object in the container for the computer on which the instance is running. An SCP is the type of Active Directory object that represents services available on the network. An SCP object records information about the service, such as connection information. An Analysis server is also registered as an SCP of the computer on which the Analysis server is running.

After registering an instance of the SQL Server relational database engine in the Active Directory, you can also register the replication publications that reside in the instance. The publications are registered as children of the instance. After registering replication publications in the Active Directory, the Create Pull Subscription Wizard supports a dialog box that allows users to search for registered publications in the Active Directory. For more information, see <u>Active Directory Services</u>.

After registering an instance of the relational database engine in the Active Directory, you can also register any databases in that instance. In SQL Server Enterprise Manager, right-click the database and select **Properties**. The **Options** tab has a check box at the bottom that controls whether the database is registered in the Active Directory. When you select the checkbox, the database is registered in the Active Directory when you close the **Properties** dialog box. After the check box is selected, the database object in the Active Directory is refreshed each time you close the **Properties** dialog box, provided the check box is selected when you open the **Properties** dialog box and remains checked when you click **OK** to close the dialog box. You can also use the **sp\_ActiveDirectory\_Obj** stored procedure to register databases from Transact-SQL scripts or applications.

You can register Analysis servers in the Active Directory. For more information, see <u>Using Active Directory with Analysis Services</u>. The SQL Server 2000 tools do not provide any facilities for browsing the Active Directory for instances of the relational database engine, Analysis servers, or relational databases. Applications can be coded to browse the Active Directory for the objects used to register these SQL Server entities.

The Active Directory class objects supported by SQL Server 2000 are defined in the Windows 2000 Active Directory schema:

Active Directory Object Name	SQL Server Entity
MS-SQL-SQLServer	An instance of SQL Server
MS-SQL-SQLPublication	A replication publication defined in an instance of SQL Server.
MS-SQL-SQLDatabase	A database in an instance of SQL Server.
MS-SQL-OLAPServer	An instance of the SQL Server Analysis server.

SQL Server 2000 makes no extensions to the definitions of these objects; SQL Server uses the objects as defined in the Windows 2000 Active Directory schema. Users can also code Active Directory Service Interfaces (ADSI) applications that browse the Active Directory for registered instances of SQL Server, Analysis servers, publications, and databases, For more information about ADSI and the structure of Active Directory schema objects, see the MSDN® Library at Microsoft Web site.

**Note** SQL Server 2000 does not use the **MS-SQL-OLAPCube**, **MS-SQL-OLAPDatabase**, or **MS-SQL-SQLRepository** class objects defined in the Windows 2000 Active Directory schema.

#### See Also

MSSQLServerADHelper Service\_mssqlserveradhelper\_service

sp\_ActiveDirectory\_SCP\_sp\_activedirectory\_scp

## **SQL Server and XML Support**

Extensible Markup Language (XML) is a hypertext programming language used to describe the contents of a set of data and how the data should be output to a device or displayed in a Web page. Markup languages originated as ways for publishers to indicate to printers how the content of a newspaper, magazine, or book should be organized. Markup languages for electronic data perform the same function for electronic documents that can be displayed on different types of electronic gear.

Both XML and the Hypertext Markup Language (HTML) are derived from Standard Generalized Markup Language (SGML). SGML is a very large, complex language that is difficult to fully use for publishing data on the Web. HTML is a more simple, specialized markup language than SGML, but has a number of limitations when working with data on the Web. XML is smaller than SGML and more robust than HTML, so is becoming an increasingly important language in the exchange of electronic data through the Web or intracompany networks.

In a relational database such as Microsoft® SQL Server<sup>™</sup> 2000, all operations on the tables in the database produce a result in the form of a table. The result set of a SELECT statement is in the form of a table. Traditional client/server applications that execute a SELECT statement process the results by fetching one row or block of rows from the tabular result set at a time and mapping the column values into program variables. Web application programmers, on the other hand, are more familiar with working with hierarchical representations of data in XML or HTML documents.

SQL Server 2000 introduces support for XML. These new features include:

- The ability to access SQL Server through a URL.
- Support for XML-Data schemas and the ability to specify XPath queries against these schemas.
- The ability to retrieve and write XML data:

- Retrieve XML data using the SELECT statement and the FOR XML clause.
- Write XML data using the OpenXML rowset provider.
- Enhancements to the Microsoft SQL Server 2000 OLE DB provider (SQLOLEDB) that allow XML documents to be set as command text and to return result sets as a stream.

#### See Also

XML and Internet Support Overview

Accessing SQL Server Using a URL

Creating XML Views Using Annotated Schemas

**Using XPath Queries** 

Retrieving and Writing XML Data

## **Managing Clients**

A client is a front-end application that uses the services provided by a server. The computer that hosts the application is referred to as the client computer. Client software enables computers to connect to an instance of Microsoft® SQL Server<sup>TM</sup> on a network.

SQL Server clients can include applications of various types, such as:

• OLE DB consumers.

These applications use the Microsoft OLE DB Provider for SQL Server or the Microsoft OLE DB Provider for ODBC to connect to and converse with instances of SQL Server. The OLE DB providers serve as intermediaries between an instance of SQL Server and client applications that consume SQL Server data as OLE DB rowsets.

• ODBC applications.

These include client utilities installed with SQL Server, such as SQL Server Enterprise Manager and SQL Query Analyzer, as well as other applications that use the SQL Server ODBC driver to connect to and converse with an instance of SQL Server.

• DB-Library clients, including the SQL Server **isql** command prompt utility and clients written to DB-Library.

Regardless of the type of application, managing a client consists mainly of configuring its connection with the server components of SQL Server. Depending on the requirements of your site, client management can range from little more than entering the name of the server computer to building a library of custom configuration entries to accommodate a diverse multiserver environment.

### How to start the Client Network Utility (Windows)

#### To start the Client Network Utility

• On the **Start** menu, point to **Programs/Microsoft SQL Server**, and then click **Client Network Utility**.

### How to display the network library version numbers (Client Network Utility)

To display the library version numbers

• Click the **Network Libraries** tab.

The network library, library file name, version, file date, and size are displayed.

# How to set DB-Library conversion preferences (Client Network Utility)

**Note** This procedure applies to Microsoft® Windows® 32-bit operating system clients.

To set the DB-Library conversion preferences

- 1. Click the **DB-Library Options** tab.
- 2. Select or clear the **Automatic ANSI to OEM conversion** check box.
- 3. Select or clear the **Use international settings** check box.

# How to add a network library configuration (Client Network Utility)

#### To add a network library configuration

- 1. Click the **Alias** tab, and then click **Add**.
- 2. In the **Add Network Library Configuration** dialog box, under **Network libraries**, select one of the network libraries.
- 3. Enter the server alias and any required parameter information for the network library selected.

# How to edit a network library configuration (Client Network Utility)

#### To edit a network library configuration

- 1. Click the **Alias** tab, and then click the network protocol configuration to edit.
- 2. Click **Edit**.
- 3. In the **Edit Network Library Configuration** dialog box, edit the information to change.

# How to delete a network library configuration (Client Network Utility)

#### To delete a network library configuration

- 1. Click the **Alias** tab, and then click the network library configuration to delete.
- 2. Click **Remove**.

# How to alias a client to an alternate pipe (Client Network Utility)

#### To alias a client to an alternate pipe

- 1. Click the **Alias** tab, and then click **Add**.
- 2. In the **Add Network Library Configuration** dialog box, click **Named Pipes**.
- 3. In the **Server alias** box, enter the server alias.
- 4. Under **Connection parameters**, in the **Pipe name** box, type the name of the alternate pipe name (for example, \\myserver\pipe\altpipe).

### How to configure a client to use the Multiprotocol Net-Library (Client Network Utility)

**Note** Before creating a Multiprotocol client configuration, make sure your computer has at least one IPC protocol loaded under Multiprotocol on the server (Named Pipes, NWLink IPX/SPX, TCP/IP, or Windows Sockets).

#### To configure a client to use the Multiprotocol Net-Library

- 1. Click the **Alias** tab, and then click **Add**.
- 2. In the Add Network Library Configuration dialog box, click Multiprotocol.
- 3. In the **Server alias** box, enter the name of the instance of Microsoft® SQL Server<sup>™</sup> listening on the Multiprotocol Net-Library.
- 4. Leave the **Additional parameters** box empty, unless the server requires specific parameters. Verify with your network administrator before entering parameters.

# How to configure a client to use TCP/IP (Client Network Utility)

#### To configure a client to use TCP/IP

- 1. Click the **General** tab, and then click **Add**.
- 2. In the **Add Network Library Configuration** dialog box, click **TCP/IP**.
- 3. In the **Server alias** box, enter the alias of the instance of Microsoft® SQL Server<sup>™</sup> listening on the Windows Sockets Net-Library.

With TCP/IP, you can also specify the server with its IP address instead of its name.

- 4. Do one of the following:
  - Select the **Dynamically determine port** check box to automatically determine the port.
  - Clear the **Dynamically determine port** check box to set the port manually, and then in the **Port number** box, type the port number.

For more information about other TCP/IP protocols that support Windows Sockets, see the TCP/IP documentation.

# How to configure a client to use the NWLink IPX/SPX network library (Client Network Utility)

To configure a client to use the NWLink IPX/SPX network library

- 1. Click the **Alias** tab, and then click **Add**.
- 2. In the **Add Network Library Configuration** dialog box, click **NWLink IPX/SPX**.
- 3. In the **Server alias** box, enter the alias of the instance of Microsoft® SQL Server<sup>™</sup> listening on the NWLink IPX/SPX Net-Library.
- 4. Under **Connection parameters**, click either **Service name** or **Network address**, and then do one of the following:
  - If you clicked **Service name**, enter the service name.

**Service name** is the Microsoft Windows NT® 4.0 or Windows® 2000 computer name under which an instance of SQL Server is running. This name is stored in the Bindery of the server computer.

• If you clicked **Network address**, enter the address (the MAC address), port (socket number), and network (NetWare network number).

### How to configure a client to use the AppleTalk network library (Client Network Utility)

To configure a client to use the AppleTalk network library

- 1. Click the **Alias** tab, and then click **Add**.
- 2. In the **Add Network Library Configuration** dialog box, click **AppleTalk**.
- 3. In the **Server alias** box, enter the name of the instance of Microsoft® SQL Server<sup>™</sup> listening on the AppleTalk Net-Library.
- 4. Under **Connection** parameters, type the AppleTalk object name and optional zone identifiers.

# How to configure a client to use the Banyan VINES network library (Client Network Utility)

To configure a client to use the Banyan VINES network library

- 1. Click the **Alias** tab, and then click **Add**.
- 2. In the **Add Network Library Configuration** dialog box, click **Banyan VINES**.
- 3. In the **Server alias** box, enter the alias of the instance of Microsoft® SQL Server<sup>™</sup> listening on the Banyan VINES Net-Library.
- 4. Under **Connection** parameters, type the service and the VINES organization. You can use the default value of MSSQL for group.

### How to configure a client to use the VIA network library (Client Network Utility)

To configure a client to use the Banyan VINES network library

- 1. Click the **Alias** tab, and then click **Add**.
- 2. In the Add Network Library Configuration dialog box, click VIA.
- 3. In the **Server alias** box, enter the alias of the instance of Microsoft® SQL Server<sup>™</sup> listening on the VIA Net-Library.
- 4. Under **Connection parameters**, type the server name and server port number.

# How to configure a client to use a nonstandard network library (Client Network Utility)

To configure a client to use a nonstandard network library

- 1. Click the **Alias** tab, and then click **Add**.
- 2. In the Add Network Library Configuration dialog box, click Other.
- 3. In the **Server alias** box, enter the alias of the instance of Microsoft® SQL Server<sup>™</sup> listening on the Net-Library you plan to install.
- 4. Under **Connection parameters**, type the file name (file must be a DLL) of the installed Net-Library. Do not enter the DLL extension.
- 5. If necessary, enter any additional information in the **Parameters** box (such as user name and password). Use comma separators between parameters.

### How to verify that SQL Server is listening on AppleTalk and can accept a client connection (Client Network Utility)

To verify that SQL Server is listening on AppleTalk and can accept a client connection

- 1. Copy the client AppleTalk Net-Library (Dbmsadsn.dll) from the \WINNT\system32 directory of the server to the same directory of a remote computer running Microsoft® Windows NT® or Microsoft Windows® 2000 Services for Macintosh.
- 2. On the remote workstation, start **SQL Server Client Configuration**.
- 3. If **AppleTalk** is listed in the **Disabled protocols** list, click **AppleTalk**, and then click **Enable**.
- 4. In the **Enabled protocols by order** list, click **AppleTalk**, and then click the **up** button until **AppleTalk** is at the top of the list.
- 5. Click **OK**.
- 6. Attempt an ISQL connection with the AppleTalk service object name. For example, at the command line, type:

isql -Usa -P -Sservicename

If you can connect with ISQL and execute queries, the server is configured properly and is accepting connections.

Note The Microsoft Win32® AppleTalk (ADSP) client side Net-

Library (Dbmsadsn.dll) is included for testing ADSP connections and troubleshooting AppleTalk connections between Macintosh clients and Microsoft SQL Server<sup>TM</sup>. This Net-Library is intended to be used only when testing a connection from a remote client to an instance of SQL Server. If you attempt to make local connections through the ADSP Net-Library to an instance of SQL Server listening on AppleTalk, you will receive the following network error: Net-Library error 11: getsockopt().

# How to check the ODBC SQL Server driver version (Windows)

**Note** You can follow these steps only if you are running the Microsoft® Windows NT® 4.0 operating system.

To check the ODBC SQL Server driver version (32-bit ODBC)

- 1. In Control Panel, double-click **ODBC Data Sources**.
- 2. Click the **Drivers** tab.

Information for the Microsoft SQL Server<sup>™</sup> entry is displayed in the **Version** column.

#### Using the Copy Database Wizard

The Copy Database Wizard allows you to copy or move databases between servers. You can move and copy databases between different instances of Microsoft® SQL Server<sup>™</sup> 2000, and you can upgrade databases from SQL Server version 7.0 to SQL Server 2000. For more information, see <u>Database</u> <u>Upgrade from SQL Server 7.0 (Copy Database Wizard)</u>.

To upgrade databases online using the Copy Database Wizard

# **Managing Security**

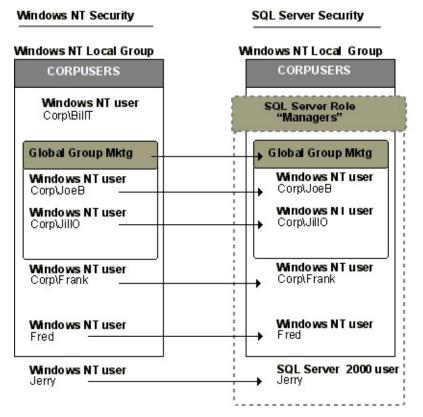
A database must have a solid security system to control which activities can be performed and which information can be viewed and modified. A solid security system ensures the protection of data, regardless of how users gain access to the database.

This section describes the security tools built into Microsoft® SQL Server<sup>™</sup> 2000 and includes information about:

- <u>Security Architecture</u>
- <u>Planning Security</u>
- <u>Creating Security Accounts</u>
- Managing Security Accounts
- <u>Managing Permissions</u>
- Advanced Security Topics
- <u>Auditing SQL Server Activity</u>

# **Security Architecture**

The architecture of a security system is based on users and groups of users. The following illustration shows how users and local and global groups in Microsoft® Windows NT® 4.0 and Windows® 2000 can map to security accounts in Microsoft SQL Server<sup>TM</sup>, and how SQL Server can handle security accounts independently of the accounts in Windows NT 4.0 and Windows 2000.



The **CORPUSERS** local group contains two users and a global group, **Mktg**, which also contains two users. SQL Server allows Windows NT 4.0 and Windows 2000 local and global groups to be used directly to organize its user accounts. Additionally, the Windows NT 4.0 users **Fred** and **Jerry**, not part of a Windows NT 4.0 group, can be added to an instance of SQL Server either directly as a Windows NT 4.0 user (**Fred** for example), or as a SQL Server user (**Jerry**).

SQL Server extends this model further with the use of roles. Roles are groups of users organized for administrative purposes, like Windows NT 4.0 or Windows 2000 groups, but are created in SQL Server when an equivalent Windows NT 4.0

or Windows 2000 group does not exist. For example, the **Managers** role contains the Windows NT 4.0 **Mktg** global group and the Windows NT 4.0 users **Frank** and **Fred**.

SQL Server also provides security at the application level through the use of individual database application roles.

For more information, see the Windows NT 4.0 or Windows 2000 documentation.

#### See Also

**Creating Security Accounts** 

### **Planning Security**

A security plan identifies which users can see which data and perform which activities in the database. To developing a security plan:

- 1. List all the items and activities in the database that must be controlled through security.
- 2. Identify the individuals and groups in the company.
- 3. Cross-reference the two lists to identify which users can see which sets of data and perform which activities in the database.

#### See Also

Single Person Security Example

Small Company Security Example

Corporate Environment Security Example

#### **Single Person Security Example**

In the simplest possible security system, a single person is responsible for all aspects of the database and will be its sole user. This hypothetical user (Tom Brown in London) must be able to:

- Create the database and its tables.
- Write programs that interface with the data.
- Load and maintain data.
- Produce reports.

The users-to-activity map for this example lists the single user and the activities he needs to perform.

User account	Activity
LONDON\tombrown	All database access

The first step in creating a security system is to add a Microsoft® SQL Server<sup>™</sup> login for **LONDON\tombrown**, allowing him access to SQL Server. Because the predefined **sysadmin** role contains all permissions necessary for this user, the **LONDON\tombrown** SQL Server login should be added as a member of the **sysadmin** role. When **LONDON\tombrown** connects to an instance of SQL Server, SQL Server calls back to Microsoft Windows NT® 4.0 or Windows® 2000 to authenticate the connection. If it is validated, the connection is accepted, and he is allowed to perform activities based on the permissions associated with the **sysadmin** role.

If Tom Brown did not have a Windows NT 4.0 or Windows 2000 login, he could be given a SQL Server login. In this case, an instance of SQL Server would need to be running under Mixed Mode, which allows users to log in under Windows NT 4.0, Windows 2000, or SQL Server logins. A login named **tombro** could be added to SQL Server independent of the Windows NT 4.0 or Windows 2000 login, and **tombro** could then be added to the **sysadmin** role. When the user logs into Windows NT 4.0 or Windows 2000 and attempts to connect to an instance of SQL Server, he must specify the **tombro** login name and password that SQL Server knows.

# **Small Company Security Example**

In a moderately complex security system, multiple people perform various tasks in the database. For example, a database administrator is responsible for the database environment: creating the database, tables, and security accounts, performing backups, and tuning the database. Two developers are responsible for writing client applications to provide an interface to the data. Managers prepare information reports from the database and so need access to all available data. The administrative staff performs customer and sales data entry and must be able to view all data.

The users-to-activity map for this example is slightly more complicated than a single user database.

User account	Activity	
LONDON\joetuck	All database access.	
LONDON\marysmith,	Full access to data and the ability to create	
LONDON\billb	procedures.	
LONDON\managers	Full access to all data.	
LONDON\admins	Full access to customer data and sales. Read-only	
	access for all other data.	

The first step in installing the security for this example is to add login rights for **LONDON\joetuck**. Then, because the **LONDON\joetuck** login requires full access, the next step is to add this user to the **sysadmin** role.

Login rights should be added for the developers, too. One way to do this is to grant individual developers (LONDON\marysmith and LONDON\billb) permissions to access data. But if another developer (or another 10 developers) joined the project, separate permissions would have to be added to each new person, a time-consuming task. A better solution is to add a SQL Server database role named **Developers**, granting permissions to access data and creating procedures to the role. When LONDON\marysmith and LONDON\billb, or accounts for other new developers, are added to the **Developers** role, their user accounts get the permissions granted to the role.

Roles are only applicable at the database level. That is, roles solve the problem

of controlling database user access. Instead of individually granting database access to 10 developers, you can create a role, add the 10 developers to it, and grant the role database access.

Finally, login rights must be added to SQL Server for **LONDON\managers** and **LONDON\admins**. When a manager connects, she is recognized as a member of an existing Microsoft Windows NT® 4.0 and Windows® 2000 group and allowed to perform activities based on the permissions granted to that group. The same is true for **LONDON\admins**.

# **Corporate Environment Security Example**

In a large corporate security system, there is a complex web of users who perform specialized, exclusive tasks.

A single person is responsible for all aspects of the database application. A few people are responsible for creating databases and tables, but they must not be allowed to see sensitive personnel information about their coworkers (or even themselves). An evening team backs up data, but these workers need not see the data, nor create tables and databases. The Personnel department must have access to general employee information, and a few select individuals in this department are the only people in the company with access to confidential and sensitive employee information. Also, customer service employees need to see but not change product specifications in response to customer inquiries.

User account	Activity
LONDON\annej	All database access
LONDON\dbadmins	Create databases
LONDON\dboperations	Perform evening backups
LONDON\personnel	Full access to general employee data
LONDON\mikebo,	Full access to confidential data
LONDON\marym,	
LONDON\billsm	
LONDON\custservice	Read-only access to product information

The users-to-activity map for this example is fairly complex.

The **LONDON**\**annej** user account must be granted login rights to Microsoft® SQL Server<sup>™</sup> and added to the **sysadmin** role because the **sysadmin** role has full permissions across the server. The **LONDON**\**dbadmins** Microsoft Windows NT® 4.0 and Windows® 2000 group user account must be added in SQL Server and granted permission to create databases. The

**LONDON**\**operations** Windows NT 4.0 group should be added also and granted only the BACKUP DATABASE permissions to allow them to perform backups.

The **LONDON**\**personnel** Windows NT 4.0 and Windows 2000 group should be added and granted the permissions to see only the nonsensitive columns in the

employees table, as well as the permissions to see other tables.

The users **LONDON\mikebo**, **LONDON\marym**, and **LONDON\billsm** are members of the **LONDON\personnel** Windows NT 4.0 group, so they already have the permissions necessary to do most of their work. However, they also need special access to the sensitive employee information columns. To meet this need, create a database role called **PersonnelSecure** in SQL Server and grant the permissions required to see the sensitive employee information. Individual users get the special permissions in SQL Server when added to the role. Or, add the special permissions to their user accounts directly.

The final step is to add an account for the **LONDON**\**custservice** Windows NT 4.0 group in SQL Server, and grant it permission to see product information.

#### **Security Levels**

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

## **Authentication Modes**

Microsoft<sup>®</sup> SQL Server<sup>™</sup> can operate in one of two security (authentication) modes:

• Windows Authentication Mode (Windows Authentication)

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

• Mixed Mode (Windows Authentication and SQL Server Authentication)

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

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

#### Windows Authentication

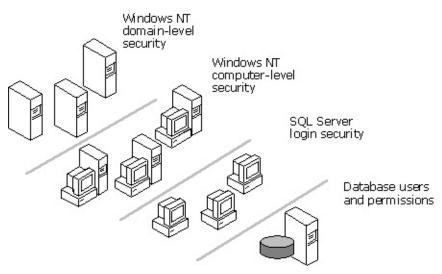
When a user connects through a Windows NT 4.0 or Windows 2000 user account, SQL Server revalidates the account name and password by calling back to Windows NT 4.0 or Windows 2000 for the information.

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

Login security integration operates over any supported network protocol in SQL

Server.

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



Windows Authentication has certain benefits over SQL Server Authentication, primarily due to its integration with the Windows NT 4.0 and Windows 2000 security system. Windows NT 4.0 and Windows 2000 security provides more features, such as secure validation and encryption of passwords, auditing, password expiration, minimum password length, and account lockout after multiple invalid login requests.

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

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

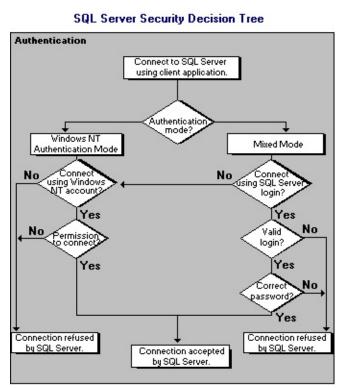
#### **SQL Server Authentication**

When a user connects with a specified login name and password from a

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

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

Application developers and database users may prefer SQL Server Authentication because they are familiar with the login and password functionality. SQL Server Authentication may also be required for connections with clients other than Windows NT 4.0 and Windows 2000 clients.



**Note** When connecting to an instance of SQL Server running on Windows NT 4.0 or Windows 2000 using Named Pipes, the user must have permission to connect to the Windows NT Named Pipes IPC, \\<computername>\**IPC\$**. If the

user does not have permission to connect, it is not possible to connect to an instance of SQL Server using Named Pipes unless either the Windows NT 4.0 or Windows 2000 **guest** account on the computer is enabled (disabled by default), or the permission "access this computer from the network" is granted to their user account.

#### To set up Windows Authentication Mode security

## **Security Account Delegation**

Security account delegation is the ability to connect to multiple servers, and with each server change, to retain the authentication credentials of the original client. For example, if a user (**LONDON**\**joetuck**) connects to ServerA, which then connects to ServerB, ServerB knows that the connection security identity is **LONDON**\**joetuck**.

To use delegation, all servers that you are connecting to must be running Microsoft® Windows® 2000, with Kerberos support enabled, and you must be using Microsoft Active Directory<sup>™</sup>, the directory service for Windows 2000. The following options in Active Directory must be specified as follows in order for delegation to work:

- The **Account is sensitive and cannot be delegated** check box must not be selected for the user requesting delegation.
- The **Account is trusted for delegation** check box must be selected for the service account of SQL Server.
- The **Computer is trusted for delegation** check box must be selected for the server running an instance of Microsoft SQL Server<sup>™</sup>.

To use security account delegation, SQL Server must have:

• A Service Principal Name (SPN) assigned by the Windows 2000 account domain administrator.

The SPN must be assigned to the service account of the SQL Server service on that particular computer. Delegation enforces mutual authentication. The SPN proves that SQL Server is verified on the particular server, at the particular socket address, by the Windows 2000 account domain administrator. You can have your domain administrator establish an SPN for SQL Server with the **setspn** utility through the Windows 2000 Resource Kit.

To create an SPN for SQL Server, enter the following code at a

command prompt:

setspn -A MSSQLSvc/Host:port serviceaccount

For example:

setspn -A MSSQLSvc/server1.redmond.microsoft.com sqlacco

For more information about the **setspn** utility, see the Windows 2000 documentation.

Before enabling delegation, consider the following:

- You must be using TCP/IP. You cannot use Named Pipes, because the SPN targets a particular TCP/IP socket. If you are using multiple ports, you must have a SPN for each port.
- You can also enable delegation by running under the **LocalSystem** account. SQL Server will self-register at service startup and automatically register the SPN. This option is easier than enabling delegation using a domain user account. However, when SQL Server shuts down, the SPNs will be unregistered for the **LocalSystem** account.

**Note** If you change service accounts in SQL Server, you need to delete any previous SPNs and create new ones.

#### Adding an SPN to SQL Server

To add an SPN on an instance of SQL Server named "myserver.microsoft.com", for an instance listening on port 1433, using service account MYDOMAIN\sqlsvc, run the following at a command prompt:

setspn -A MSSQLSvc/myserver.microsoft.com:1433 sqlsvc

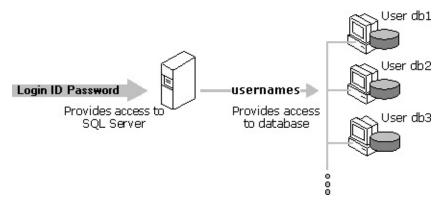
You cannot use the Netbios name. You must use the fully qualified DNS name. You cannot specify the domain qualifier for the service account. You must use only the account name. To change and use the **LocalSystem** account, enter the following code at a command prompt to delete the previously registered SPN :

setspn -D MSSQLSvc/myserver.microsoft.com:1433 sqlsvc

For more information about security account delegation, see the Windows 2000 documentation.

#### **Permissions Validation**

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



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

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

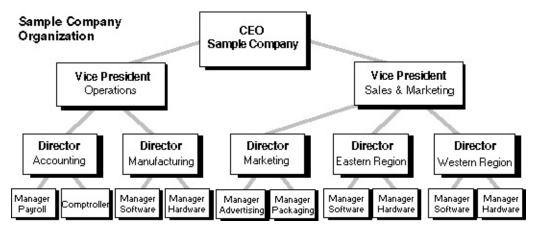
# **Hierarchical Security**

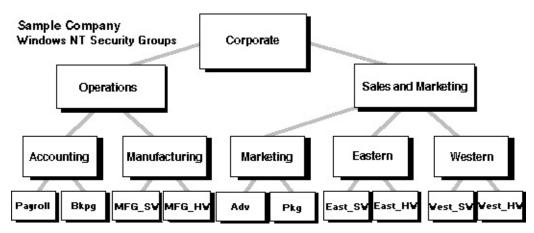
The security environment in Microsoft® SQL Server<sup>™</sup> is stored, managed, and enforced through a hierarchical system of users. To simplify the administration of many users, SQL Server uses groups and roles:

- A group is an administrative unit within Microsoft Windows NT® 4.0 and Windows® 2000 that contains Windows NT 4.0 and Windows 2000 users or other groups.
- A role is an administrative unit within SQL Server that contains SQL Server logins, Windows NT 4.0 and Windows 2000 logins, groups, or other roles.

Arranging users into groups and roles makes it easier to grant or deny permissions to many users at once. The security settings defined for a group are applied to all members of that group. When a group is a member of a higherlevel group, all members of the group inherit the security settings of the higherlevel group, in addition to the security settings defined for the group itself or user accounts.

The organizational chart of the security system often corresponds to the organizational chart of a company.





These two organizational charts are largely compatible, but there is one common rule for a company's organizational hierarchy that does not apply to the security model: an individual reports only to one manager. This rule implies that an employee can fall into only a single branch of the hierarchical model, as shown in the diagram.

The requirements of a database security system go beyond this one-manager limitation; employees belong to security groups that do not fall within the strict organizational plan of the company. For example, administrative staff exists in every branch of the company and require security permissions regardless of their organizational branch. To support this broader model, the security system in Windows NT 4.0, Windows 2000, and SQL Server allows groups to be defined across a hierarchy. An **Administrative** group can be created to contain administrative employees for every branch of the company from the **Corporate** group to the **Payroll** group.

This hierarchical system of security groups simplifies management of security settings. It allows security settings to be applied collectively to all group members, without having to be defined redundantly for each person. The hierarchical model also accommodates security settings applied only to a single user.

## **Creating Security Accounts**

Each user must gain access to an instance of Microsoft® SQL Server<sup>™</sup> through a login account that establishes the user's ability to connect (authentication). This login then has to be mapped to a SQL Server user account, which is used to control activities performed in the database (permissions validation). Therefore, a single login is mapped to one user account created in each database the login is accessing. If no user account exists in a database, the user cannot access the database even though the user may be able to connect to an instance of SQL Server.

The login is created in Microsoft Windows NT® 4.0 or Windows® 2000 rather than in SQL Server. This login is then granted permission to connect to an instance of SQL Server. The login is granted access within SQL Server.

## **Security Rules**

Microsoft® SQL Server<sup>™</sup> logins, users, roles, and passwords can contain from 1 through 128 characters, including letters, symbols, and digits, (for example **Andrew-Fuller**, **Margaret Peacock**, or **13&#57abc**). Therefore, Microsoft Windows NT® 4.0, Microsoft Windows® 2000, or Microsoft Windows 98 user names can be used as SQL Server logins.

However, because logins, user names, roles, and passwords are often used in Transact-SQL statements, certain symbols must be delimited with double quotation marks ("), or square brackets ([]). Use delimiters in Transact-SQL statements when the SQL Server login, user, role, or password:

- Contains, or begins with, a space character.
- Begins with the **\$** or **@** character.

**Note** It is not necessary to specify delimiters when entering logins, users, roles, and passwords into the text boxes of the SQL Server graphical client tools, such as SQL Server Enterprise Manager.

Additionally, a SQL Server login, user, or role cannot:

- Contain a backslash (\) character, unless referring to an existing Windows NT 4.0 or Windows 2000 user or group. The backslash separates the Windows NT 4.0 or Windows 2000 computer or domain name from the user name.
- Already exist in the current database (or **master**, for logins only).
- Be NULL, or an empty string ("").

#### See Also

**Delimited Identifiers** 

## Adding a Windows User or Group

Microsoft® Windows NT® 4.0 and Windows® 2000 accounts (users or groups) must be granted permissions to connect to an instance of Microsoft SQL Server<sup>™</sup> before they can access a database. If all members of a Windows NT 4.0 or Windows 2000 group will be connecting to an instance of SQL Server, you can grant permission to the group as a whole. Managing group permissions is much easier than managing permissions for individual users. If the group should not be granted permission collectively, grant permission to connect to an instance of SQL Server for each individual Windows NT 4.0 or Windows 2000 user.

#### Users

When granting a Windows NT 4.0 or Windows 2000 user access to connect to an instance of SQL Server, specify the Windows NT 4.0 or Windows 2000 domain or computer name to which the user belongs, followed by a backslash, and then the user. For example, to grant access to the Windows NT 4.0 or Windows 2000 user **Andrew**, in the Windows NT 4.0 or Windows 2000 domain **LONDON**, specify **LONDON**\**Andrew** as the user name.

#### Local and Global Groups

There are several types of Windows NT 4.0 and Windows 2000 groups, including global and local:

- Global groups contain user accounts from the Windows NT 4.0 or Windows 2000 domain in which they are created. Global groups cannot contain other groups or users from other domains and cannot be created on a computer running Microsoft Windows NT 4.0 Workstation or Microsoft Windows 2000 Professional.
- Local groups can contain user accounts and global groups from the domain in which they are created and in any trusted domain. Local groups cannot contain other local groups.

Additionally, Windows NT 4.0 and Windows 2000 have predefined, built-in local groups (for example, **Administrators**, **Users**, and **Guests**).

When granting a Windows NT 4.0 or Windows 2000 local or global group access to connect to an instance of SQL Server, specify the domain or computer name the group is defined on, followed by a backslash, and then the group name. For example, to grant access to a global group called **SQL\_Users**, in the **LONDON** domain, specify **LONDON**\**SQL\_Users** as the group name.

To grant access to a Windows NT 4.0 or Windows 2000 built-in, local group, specify BUILTIN instead of the domain or computer name. To grant access to the built-in Windows NT 4.0 and Windows 2000 local group **Administrators**, specify **BUILTIN**\**Administrators** as the group name.

For more information about these accounts, see the Windows NT 4.0 and Windows 2000 documentation.

#### To grant a Windows user or group login access to SQL Server

⊞ <u>Transact-SQL</u>

# Granting a Windows User or Group Access to a Database

To obtain access to a Microsoft® SQL Server<sup>™</sup> database, a Microsoft Windows NT® 4.0 and Windows® 2000 user or group must have a corresponding user account in each database they need to access. Additionally, permissions must be applied to this user account.

Although possible, it is not necessary to add an individual user account in a database for each Windows NT 4.0 and Windows 2000 user in a Windows NT 4.0 and Windows 2000 group whose members all perform the same activities. Accounts can be added for groups rather than for each individual member. When the group members need to work in a database, they are granted access through their membership in the Windows NT 4.0 and Windows 2000 group; there is not a specific account for individual users within the group. For example, a Windows NT 4.0 and Windows 2000 group London\Managers contains the Windows NT 4.0 and Windows 2000 user London\JoeB. The SQL Server system administrator grants login access only to London\Managers. The owner of database Accounts grants only London\Managers permission to access Accounts. Although London\JoeB does not have explicit permission granted to connect to an instance of SQL Server or to access Accounts, he can connect to the instance of SQL Server and access Accounts due to his membership in London\Managers.

Add individual Windows NT 4.0 and Windows 2000 users to a database only if the user performs activities different from other members of any Windows NT 4.0 or Windows 2000 group (for example, special database administrative duties).

**Note** Users who are granted access to an instance of SQL Server through their memberships in a Windows NT 4.0 or Windows 2000 group do not have entries for their individual Windows NT 4.0 or Windows 2000 user accounts in the system tables. However, an entry is created for their individual user accounts if they create objects, such as a table or a stored procedure, in a SQL Server database.

#### To grant a Windows user or group access to a database

⊞ <u>Transact-SQL</u>

## Adding a SQL Server Login

Add Microsoft® SQL Server<sup>™</sup> login accounts that allow a connection by means of a specified login name and password, rather than through a Microsoft Windows NT® 4.0 or Windows® 2000 user or group account, if:

- SQL Server is configured to operate in Mixed Mode.
- An instance of SQL Server is running on Microsoft Windows 98.

Adding SQL Server logins is required:

- For compatibility with applications containing data imported from other databases vendors.
- For applications designed to work with general users who do not have Windows NT 4.0 or Windows 2000 accounts.
- To connect to an instance of SQL Server running on Windows 98 because Windows Authentication is not available on Windows 98.

#### To add a SQL Server login

⊞ <u>Transact-SQL</u>

## System Administrator (sa) Login

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 changed. Although **sa** is a built-in administrator login, do not use it routinely. Instead, make system administrators members of the **sysadmin** fixed server role, and have them log on using their own logins. Use **sa** only when there is no other way to log in to an instance of Microsoft® SQL Server<sup>TM</sup> (for example, when other system administrators are unavailable or have forgotten their passwords).

**Note** When SQL Server is installed, SQL Server Setup prompts you to change the **sa** login password if you request Mixed Mode authentication. It is recommended that the password be assigned immediately to prevent unauthorized access to an instance of SQL Server using the **sa** login.

#### See Also

Assigning an sa Password

## **Granting a SQL Server Login Access to a Database**

Add a Microsoft® SQL Server<sup>™</sup> user account to each database for each SQL Server login that requires access to the database. If a user is not created in the database, the SQL Server login cannot access the database.

To grant a SQL Server login access to a database, the SQL Server login must already exist. Furthermore, SQL Server logins must be granted access to a database one at a time.

#### To grant a SQL Server login access to a database

⊞ <u>Transact-SQL</u>

## **Database Owner (dbo)**

The **dbo** is a user that has implied permissions to perform all activities in the database. Any member of the **sysadmin** fixed server role who uses a database is mapped to the special user inside each database called **dbo**. Also, any object created by any member of the **sysadmin** fixed server role belongs to **dbo** automatically.

For example, if user **Andrew** is a member of the **sysadmin** fixed server role and creates a table **T1**, **T1** belongs to **dbo** and is qualified as **dbo.T1**, not as **Andrew.T1**. Conversely, if **Andrew** is not a member of the **sysadmin** fixed server role but is a member only of the **db\_owner** fixed database role and creates a table **T1**, **T1** belongs to **Andrew** and is qualified as **Andrew.T1**. The table belongs to **Andrew** because he did not qualify the table as **dbo.T1**.

The **dbo** user cannot be deleted and is always present in every database.

Only objects created by members of the **sysadmin** fixed server role (or by the **dbo** user) belong to **dbo**. Objects created by any other user who is not also a member of the **sysadmin** fixed server role (including members of the **db\_owner** fixed database role):

- Belong to the user creating the object, not **dbo**.
- Are qualified with the name of the user who created the object.

#### See Also

Delimited Identifiers

## **Database Object Owner**

A user who creates a database object (a table, index, view, trigger, function, or stored procedure) is called a database object owner. Permission to create database objects must be given by the database owner or system administrator. However, after these permissions are granted, a database object owner can create an object and grant other users permission to use that object.

Database object owners have no special login IDs or passwords. The creator of a database object is granted all permissions implicitly but must give explicit permissions to other users before they can access the object.

#### **Referencing database objects**

When users access an object created by another user, the object should be qualified with the name of the object owner; otherwise, Microsoft® SQL Server<sup>™</sup> may not know which object to use because there could be many objects of the same name owned by different users. If an object is not qualified with the object owner when it is referenced (for example, **my\_table** instead of **owner.my\_table**), SQL Server looks for an object in the database in the following order:

- 1. Owned by the current user.
- 2. Owned by **dbo**.

If the object is not found, an error is returned.

For example, user **John** is a member of the **db\_owner** fixed database role, but not the **sysadmin** fixed server role, and creates table **T1**. All users, except **John**, who want to access **T1** must qualify **T1** with the user name **John**. If **T1** is not qualified with the user name **John**, SQL Server first looks for a table named **T1** owned by the current user and then owned by **dbo**. If the current user and **dbo** do not own a table named **T1**, an error is returned. If the current user or **dbo** owns another table named **T1**, the other table named **T1**, rather than **John.T1**, is used. If a database object owner must be removed from a database, the owned objects must be dropped first or their ownership transferred to another user.

**Note** SQL Server allows a role or Microsoft Windows NT® 4.0 or Windows® 2000 group to be specified as the owner of an object. For example, to create the table **group\_table** owned by the Windows NT 4.0 or Windows 2000 group **LONDON\Users**, specify **[LONDON\Users].group\_table** as the qualified table name. All members of the **LONDON\Users** group have database object owner permissions on **group\_table**.

#### See Also

Delimited Identifiers sp\_changeobjectowner

#### guest User

The **guest** user account allows a login without a user account to access a database. A login assumes the identity of the **guest** user when both of the following conditions are met:

- The login has access to an instance of Microsoft® SQL Server<sup>TM</sup> but does not have access to the database through his or her own user account.
- The database contains a **guest** user account.

Permissions can be applied to the **guest** user as if it were any other user account. The **guest** user can be deleted and added to all databases except **master** and **tempdb**, where it must always exist. By default, a **guest** user account does not exist in newly created databases.

For example, to add a **guest** user account to a database named **Accounts**, run the following code in SQL Query Analyzer:

USE Accounts GO EXECUTE sp\_grantdbaccess guest

#### To grant a SQL Server login access to a database

∃ <u>Transact-SQL</u>

## **Creating User-Defined SQL Server Database Roles**

Create Microsoft® SQL Server<sup>™</sup> database roles when a group of users needs to perform a specified set of activities in SQL Server and one of the following is true:

- There is no applicable Microsoft Windows NT® 4.0 or Windows® 2000 group.
- You do not have permissions to manage Windows NT 4.0 or Windows 2000 user accounts.

**Note** Avoid deep levels of nested roles because this can affect performance.

For example, a company may form a Charity Event Committee involving employees from different departments and from several different levels in the organization. These employees need access to a special project table in the database. There is no existing Windows NT 4.0 or Windows 2000 group that includes just these employees, and there is no other reason to create one in Windows NT 4.0 or Windows 2000. A custom SQL Server database role, **CharityEvent**, can be created for this project and individual Windows NT 4.0 and Windows 2000 users added to the database role. When permissions are applied, the users in the database role gain table access. Permissions for other database activities are not affected, and the **CharityEvent** users are the only ones who can work with the project table.

SQL Server roles exist within a database and cannot span more than one database.

The advantages of using database roles include:

- For any user, more than one database role can be active at any time.
- SQL Server roles can contain Windows NT 4.0 or Windows 2000 groups and users and SQL Server users and other roles, provided that all users, groups, and roles exist in the current database.

- A user can belong to more than one role in the same database.
- A scalable model is provided for setting up the correct level of security within a database.

**Note** A database role is owned by either the user explicitly specified as the owner when the role is created, or the user who created the role when no owner is specified. The owner of the role determines who can be added or removed from the role. However, because a role is not a database object, multiple roles of the same name in the same database owned by different users cannot be created.

#### To create a SQL Server database role

⊞ <u>Transact-SQL</u>

## Adding a Member to a SQL Server Database Role

When you add a new user account in Microsoft® SQL Server<sup>™</sup> or change the permissions of an existing user, you can add the user to a SQL Server database role rather than applying permissions directly to the account. Roles can simplify security administration in databases with a large number of users or with a complex security system.

SQL Server users, Microsoft Windows NT® 4.0 or Windows® 2000 users and groups, and other SQL Server database roles all can be added as a member of a role. Because a role is restricted to a single database and cannot be added from one database to another, you can add users, groups, and roles known only to that database.

**Note** When you add a Windows NT 4.0 or Windows 2000 login without a user account in the database to a SQL Server database role, SQL Server creates a user account in the database automatically, even if that Windows NT 4.0 or Windows 2000 login cannot otherwise access the database.

A user account can be a member of any number of roles within the same database and can hold permissions appropriate to each role. For example, a SQL Server user can be a member of the **admin** role and the **users** role for the same database, with each role granting different permissions. The permission on an object granted to a member of more than one role are the cumulative permissions of the roles. However, a denied permission in one role has precedence over the same permission granted in another role. For example, the **admin** role may grant access to a table, whereas the **users** role denies access to the same table. A member of both roles is denied access to the table because denied access is more restrictive and has precedence.

Users to be added to a user-defined database role must already have permission to access the database containing the user-defined role.

#### To add a member to a SQL Server database role

**⊡** <u>Transact-SQL</u>

## Adding a Member to a Predefined Role

The security mechanism in Microsoft® SQL Server<sup>™</sup> includes several predefined roles with implied permissions that cannot be granted to other user accounts. If you have users who require these permissions, you must add their accounts to these predefined roles. The two types of predefined roles are fixed server and fixed database.

#### **Fixed Server Roles**

Fixed server roles, which cannot be created, are defined at the server level and exist outside of individual databases. To add a user to a fixed server role, the user must have a SQL Server or Microsoft Windows NT® 4.0 or Windows® 2000 login account. Any member of a fixed server role can add other logins.

**IMPORTANT** Windows NT 4.0 or Windows 2000 users who are members of the **BUILTIN**\**Administrators** group are members of the **sysadmin** fixed server role automatically.

Fixed server role	Description
sysadmin	Performs any activity in SQL Server. The permissions of this role span all of the other fixed server roles.
serveradmin	Configures server-wide settings.
setupadmin	Adds and removes linked servers, and executes some system stored procedures, such as <b>sp_serveroption</b> .
securityadmin	Manages server logins.
processadmin	Manages processes running in an instance of SQL Server.
dbcreator	Creates and alters databases.
diskadmin	Manages disk files.
bulkadmin	Executes the BULK INSERT statement.

The following table describes the fixed server roles.

The **securityadmin** has permission to execute the **sp\_password** stored procedure for all users other than members of the **sysadmin** role.

The **bulkadmin** fixed server role has permission to execute BULK INSERT statements. Members of the **bulkadmin** role can add other logins to the role, as all members of any given fixed server role can do. However, due to the security implications associated with executing the BULK INSERT statement (the BULK INSERT statement requires read access to any data on the network and machine the server is running on), it may not be desirable for members of the **bulkadmin** role to grant permission to others. The **bulkadmin** role provides members of the **sysadmin** fixed server role with a method to delegate tasks requiring execution of the BULK INSERT statement, without granting users **sysadmin** rights. Members of the **bulkadmin** role are allowed to execute the BULK INSERT statement, but they still must have the INSERT permission on the table on which you wish to insert data.

#### To add a member to a fixed server role

⊞ <u>Transact-SQL</u>

## public Role

The **public** role is a special database role to which every database user belongs. The **public** role:

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

## **Using the Create Login Wizard**

Although the steps required to grant login access to Microsoft® SQL Server<sup>TM</sup> and a database can be performed separately, the Create Login Wizard can simplify the process. The Create Login Wizard allows you to:

- Choose which authentication mode to use to connect to an instance of SQL Server (Windows Authentication Mode or Mixed Mode).
- Add a Microsoft Windows NT® 4.0, Windows® 2000 or SQL Server login.
- Add a Windows NT 4.0, Windows 2000 or SQL Server user to a fixed server role.
- Add a Windows NT 4.0, Windows 2000 or SQL Server user to one or more databases, thereby granting the user access to those databases.

## To grant SQL Server login access to a user by using the Create Login Wizard

## **Managing Security Accounts**

After security accounts have been added to Microsoft® SQL Server<sup>™</sup>, you can modify them as business needs change. This usually involves viewing, modifying, and removing the security accounts in the database to fit the needs of your business.

## **Viewing Logins**

View Microsoft® SQL Server<sup>™</sup> logins to determine if a user or Microsoft Windows NT® 4.0 or Windows® 2000 group has permission to connect to an instance of SQL Server, and to identify which databases the login can access. Also, view a login before removing it to see which database users must be removed; it is not possible to remove a login without first removing the associated users.

You can view:

- Users in each database associated with the login.
- Default database and language the login uses when the user first connects to an instance of SQL Server.
- Windows NT 4.0 or Windows 2000 security identifier (SID).

**Note** It is not possible to view the password of any login unless the password is NULL. Passwords are encrypted when stored in SQL Server.

#### To view a SQL Server login or Windows user or group

∃ <u>Transact-SQL</u>

## **Modifying Logins**

After a login has been created, it may be necessary to change the password, default database, or default language. For example, a user may forget her password, want to change the password for security reasons, need to use a different database on a regular basis, or need to see messages in a different language.

**Note** If a user forgets a password, a member of the **sysadmin** or **securityadmin** fixed server role can change the password without knowing the original password. A user cannot change a password if he has forgotten it. Members of the **securityadmin** role cannot change the password of members of the **sysadmin** role.

#### To change the password of a SQL Server login

∃ <u>Transact-SQL</u>

## **Removing Logins and Users**

The process of deactivating security accounts (for example, when an employee leaves a company) is similar to the process of adding a new user. Update the security mechanism in Microsoft® Windows NT® 4.0 or Windows® 2000 by first removing the user's Windows NT 4.0 or Windows 2000 user account. If the user has a Microsoft SQL Server<sup>™</sup> user account, removed it from SQL Server along with any SQL Server database roles specifically defined for that user. Finally, remove any SQL Server login.

Removing a SQL Server user or Windows NT 4.0 or Windows 2000 user or group from a SQL Server database automatically removes the permissions defined for the user or group and prevents that user from using the database under the old security account. The permissions do not have to be removed separately. However, it is not possible to remove a user from SQL Server if that user currently owns objects (tables, procedures, or views) within a database. If the user owns objects, then either drop those objects before removing the user or transfer ownership to another existing user by using the **sp\_changeobjectowner** system stored procedure.

Removing a user does not remove a login automatically, so it does not prevent the user from connecting to an instance of SQL Server. After being removed, the user can log in to the databases only through the **guest** account and perform activities under those permissions. To prevent a user from connecting to an instance of SQL Server, remove his or her login.

If a linked server login is set up but is no longer required, remove it to prevent unauthorized access to the linked server and to keep the security system as simple as possible.

#### To remove a user or group from a database

∃ <u>Transact-SQL</u>

### **Denying Login Access to Windows Accounts**

When a Microsoft® Windows NT® 4.0 or Windows® 2000 user belongs to a Windows NT 4.0 or Windows 2000 group that has a login account in Microsoft SQL Server<sup>™</sup>, the user is allowed to connect through the group login. However, there may be times when such users or groups need to be prevented from connecting to an instance of SQL Server. You can deny login access to any Windows NT 4.0 or Windows 2000 user or group. Users cannot connect to an instance of SQL Server if their user account, or any group in which they are a member, has been denied login access.

#### To deny login access to a Windows user or group

⊞ <u>Transact-SQL</u>

# **Viewing Roles**

When creating and using a database, you may need to find information about a Microsoft® SQL Server<sup>TM</sup> database role or a fixed server role. For example, you may need to see which roles exist in the current database, or list the fixed server roles.

#### To view the roles defined in the current database

∃ <u>Transact-SQL</u>

# **Viewing and Modifying Role Memberships**

While using a database, you may need to list the members of a database role or fixed server role. Or, when a Microsoft® SQL Server<sup>™</sup> user no longer needs the permissions from a user-defined, fixed database or server role of which she is a member, you may want to remove the user from the role to keep the security system as simple as possible.

#### To view the members of a database role

⊞ <u>Transact-SQL</u>

## **Removing a SQL Server Database Role**

The changing security requirements of a database can render a Microsoft® SQL Server<sup>™</sup> database role obsolete. Remove roles when you have removed all users and are certain that the role and its permissions will not be required in the future. Empty roles can be saved if the permissions may be required for a new user. However, from an administrative perspective, it is much easier to work with a security system that is not cluttered with unnecessary security roles. SQL Server operates faster with a simpler security system, although it is will not be a problem unless there are an extremely large number of roles.

Note It is not possible to remove fixed server roles or fixed database roles.

#### To remove a SQL Server role

⊞ <u>Transact-SQL</u>

## **Viewing Database Users**

Viewing a Microsoft<sup>®</sup> SQL Server<sup>™</sup> user account in a database shows:

- The roles of which the user is a member.
- The SQL Server login associated with the user.
- The default database.

Use this information to understand how the user fits into the security system of the database.

#### To view a database user

∃ <u>Transact-SQL</u>

# **Managing Permissions**

When users connect to an instance of Microsoft® SQL Server<sup>™</sup>, the activities they can perform are determined by the permissions granted to:

- Their security accounts.
- The Microsoft Windows NT® 4.0 or Windows® 2000 groups or role hierarchies to which their security accounts belong.

The user must have the appropriate permissions to perform any activity that involves changing the database definition or accessing data.

Managing permissions includes granting or revoking user rights to:

- Work with data and execute procedures (object permissions).
- Create a database or an item in the database (statement permissions).
- Utilize permissions granted to predefined roles (implied permissions).

#### **Object Permissions**

Working with data or executing a procedure requires a class of permissions known as object permissions:

- SELECT, INSERT, UPDATE, and DELETE statement permissions, which can be applied to the entire table and view.
- SELECT and UPDATE statement permissions, which can be selectively applied to individual columns of a table or view.
- SELECT permissions, which may be applied to user-defined functions.

- INSERT and DELETE statement permissions, which affect the entire row, and therefore can be applied only to the table and view and not to individual columns.
- EXECUTE statement permissions, which affect stored procedures and functions.

### **Statement Permissions**

Activities involved in creating a database or an item in a database, such as a table or stored procedure, require a different class of permissions called statement permissions. For example, if a user must be able to create a table within a database, then grant the CREATE TABLE statement permission to the user. Statement permissions, such as CREATE DATABASE, are applied to the statement itself, rather than to a specific object defined in the database.

Statement permissions are:

- BACKUP DATABASE
- BACKUP LOG
- CREATE DATABASE
- CREATE DEFAULT
- CREATE FUNCTION
- CREATE PROCEDURE
- CREATE RULE
- CREATE TABLE

• CREATE VIEW

### **Implied Permissions**

Implied permissions control those activities that can be performed only by members of predefined system roles or owners of database objects. For example, a member of the **sysadmin** fixed server role inherits automatically full permission to do or see anything in a SQL Server installation.

Database object owners also have implied permissions that allow them to perform all activities with the object they own. For example, a user who owns a table can view, add, or delete data, alter the table definition, or control permissions that allow other users to work with the table.

### See Also

BACKUP DATABASE BACKUP LOG CREATE DATABASE CREATE DEFAULT CREATE FUNCTION CREATE PROCEDURE CREATE RULE CREATE TABLE CREATE VIEW

## **Granting Permissions**

Grant statement and object permissions that allow a user account to:

- Perform activities or work with data in the current database.
- Restrict them from activities or information not part of their intended function.

For example, you may be inclined to grant SELECT object permission on the **payroll** table to all members of the **personnel** role, allowing all members of **personnel** to view **payroll**. Months later, you may overhear members of **personnel** discussing management salaries, information not meant to be seen by all **personnel** members. In this situation, grant SELECT access to **personnel** for all columns in **payroll** except the **salary** column.

**Note** It is possible to grant permissions only to user accounts in the current database, for objects in the current database. If a user needs permissions to objects in another database, create the user account in the other database, or grant the user account access to the other database, as well as the current database. System stored procedures are the exception because EXECUTE permissions are already granted to the **public** role, which allows everyone to execute them. However, after EXECUTE has been issued, the system stored procedures check the user's role membership. If the user is not a member of the appropriate fixed server or database role necessary to run the stored procedure, the stored procedure will not continue.

#### To allow access by granting permissions (on an object)

∃ <u>Transact-SQL</u>

# **Denying Permissions**

Microsoft® SQL Server<sup>™</sup> allows Microsoft Windows NT® 4.0 or Windows® 2000 users and groups, SQL Server users, and SQL Server database roles to be members of other roles. This results in a hierarchical security system that allows permissions to be applied through several levels of roles and members. But there may be times when you want to limit the permissions of a user or role. Denying permissions on a user account:

- Removes permission granted previously to the user, group, or role.
- Deactivates permission inherited from another role(s).
- Ensures that a user, group, or role will not inherit permission from a higher level group or role in the future.

For example, you may need to provide all tenured employees in your company with access to several tables in a database, with the exception of a few new employees scattered throughout the organization who you want to prevent from seeing the **CorporateSecrets** table.

Create a role for each department in the company and add all employees to their department role. Then create a company-wide **Corporate** role, to which you add each of the individual department roles and grant permissions to view the tables. At this point, every employee in the company can see all the tables because each inherits permission from the **Corporate** role through his department roles.

To selectively prevent employees from seeing **CorporateSecrets**, create a **Nonsecure** role, and add the individual employees who should not see the table. When you deny permission to view **CorporateSecrets** to **Nonsecure**, this access is removed from all members of **Nonsecure**, while the rest of the employees in the company are not affected.

You also can deny permissions to an individual user. In the previous example, a nonemployee may have a Windows NT 4.0 or Windows 2000 account while working on a short-term project in the database. You can deny the permissions to see **CorporateSecrets** to his individual user account without creating a SQL

Server database role for the purpose.

**Note** You can deny permissions to user accounts only in the current database, for objects in the current database.

#### To prevent access by denying permissions (on an object)

⊞ <u>Transact-SQL</u>

# **Revoking Permissions**

You can revoke a permission that has been granted or denied previously. Revoking is similar to denying in that both remove a granted permission at the same level. However, although revoking a permission removes a granted permission, it does not prevent the user, group, or role from inheriting a granted permission from a higher level. Therefore, if you revoke permission for a user to view a table, you do not necessarily prevent the user from viewing the table because permission to view the table was granted to a role to which he belongs.

For example, removing SELECT access on the **Employees** table from the **HumanResources** role revokes permission so that **HumanResources** can no longer use the table. If **HumanResources** is a member of the **Administration** role. If you later grant SELECT permission on **Employees** to **Administration**, members of **HumanResources** can see the table through their membership in **Administration**. However, if you deny permission to **HumanResources**, the permission is not inherited if later granted to **Administration** because the deny permission cannot be undone by a permission at a different level.

Similarly, it is also possible to remove a previously denied permission by revoking the deny for the permission. However, if a user has other denied permissions at the group or role level, then the user still is denied access.

**Note** You can revoke permissions to user accounts only in the current database, for objects in the current database.

#### To revoke permissions on an object

⊞ <u>Transact-SQL</u>

# **Resolving Permission Conflicts**

The permissions granted to a group or role are inherited by members of that group or role. Although a user may have permission granted or revoked at one level, conflicting permissions at a higher level (for example, due to role membership) can prevent or allow a user access to an object.

### Deny

A denied permission always takes precedence. Denied permission at any level (user, group, or role) denies the permission on the object regardless of existing granted or revoked permissions for that user. For example, if user **John**, who as a member of the **sales** role is granted SELECT permissions on the **customer** table, is explicitly denied SELECT permissions on the **customer** table, he can no longer access it. Similarly, if the **sales** role is denied access to **customer**, but **John** is granted access, he is denied access.

**Note** Microsoft® SQL Server<sup>™</sup> always processes denied permissions first. If you deny permissions to **public**, you prevent anyone from accessing an object, including the issuer of the DENY statement.

### Revoke

A revoked permission removes only the granted or denied permission at the level revoked (user, group, or role). The same permission granted or denied at another level such as a group or role containing the user, group, or role still applies. For example, if the **sales** role is granted SELECT permissions on the **customer** table, and **John** (a member of **sales**) is explicitly revoked SELECT permissions on the **customer** table, he still can access the table because of his membership in the **sales** role. To prevent **John** from accessing the **customer** table, do one of the following:

- Revoke permission (assuming no other permissions have been granted elsewhere).
- Deny permission to the **sales** role (preventing all members of **sales** from accessing the table).

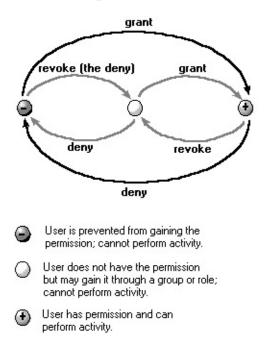
• Explicitly deny **John** SELECT permissions on **customer**.

### Grant

A granted permission removes the denied or revoked permission at the level granted (user, group, or role). The same permission denied at another level such as group or role containing the user still applies. However, although the same permission revoked at another level still applies, it does not prevent the user from accessing the object. For example, if **John** is already explicitly denied access to **customer**, has his access to **sales**, revoked, and then is explicitly granted access to **customer**, he now can access **customer** because the deny is removed. The revoke permission for **sales** joined with the granted permission for **John** gives **John** a granted permission overall.

Therefore, a user receives the union of all the permissions granted, denied, or revoked on an object, with any denied permissions taking precedence over the same permissions granted or revoked at another level.

The following diagram shows how the three permission management activities affect the state of a permission for a user account.



State Diagram for a Permission

### **Database Access vs. Object Access**

As an example of a permission conflict, a Microsoft Windows NT® 4.0 user **LONDON\joe** belongs to the **LONDON\clerks** and **LONDON\secretaries** Windows NT 4.0 groups. **LONDON\joe** can log in to an instance of SQL Server because the **LONDON\clerks** group has been granted permissions to connect to an instance of SQL Server. Additionally, **LONDON\joe** can access the **secrets** database because the **LONDON\secretaries** group has been granted permissions to access the database.

**Note** At this point there is no specific entry in the SQL Server system tables, **sysusers** and **sysxlogins**, for **LONDON**\**joe**. These system tables contain only entries for the **LONDON**\**clerks** and **LONDON**\**secretaries** groups.

**LONDON**\joe creates a table, joetable, in the secrets database. At this point, a new entry is created in the sysusers table for LONDON\joe specifying him as the object owner but not granting him database access. If LONDON\joe is dropped from the LONDON\secretaries group, he can no longer access the secrets database, although he owns an object, joetable, in the database.

### See Also

Adding a Windows NT User or Group

# **Permissions for User-Defined Functions**

Functions are subroutines made up of one or more Transact-SQL statements that can be used to encapsulate code for reuse. Microsoft® SQL Server<sup>™</sup> 2000 allows users to create their own user-defined functions.

User-defined functions are managed through the following statements:

- CREATE FUNCTION, which creates a user-defined function.
- ALTER FUNCTION, which modifies user-defined functions.
- DROP FUNCTION, which drops user-defined functions.

Each fully qualified user-defined function name (*database\_name.owner\_name.function\_name*) must be unique.

You must have been granted CREATE FUNCTION permissions to create, alter, or drop user-defined functions. Users other than the owner must be granted EXECUTE permission on a function (if the function is scalar-valued) before they can use it in a Transact-SQL statement. If the function is table-valued, the user must have SELECT permissions on the function before referencing it. If a CREATE TABLE or ALTER TABLE statement references a user-defined function in a CHECK constraint, a DEFAULT clause, or a computed column, the table owner must also own the function. If the function is being schema-bound, you must have REFERENCE permission on tables, views, and functions referenced by the function.

REFERENCE permissions can be granted through the GRANT statement to views and user-defined functions in addition to tables.

### See Also

**User-Defined Functions** 

# **Using Ownership Chains**

Views and stored procedures provide a secondary method of giving users access to data and the ability to perform activities. They provide users with access to underlying items in the database and bypass the permissions defined directly for specific objects and statements.

Views can depend on other views or tables. Procedures can depend on other procedures, views, or tables. These dependencies can be thought of as an ownership chain. Ownership chains only apply to SELECT, INSERT, UPDATE, and DELETE statements.

Typically, the owner of a view also owns the underlying objects (other views or tables), and the owner of a stored procedure often owns all the referenced procedures, tables, and views. Also, views and underlying objects are usually all in the same database, as are stored procedures and all the objects referenced. When temporary objects are created within a stored procedure, they are owned by the procedure owner and not by the user currently executing the procedure.

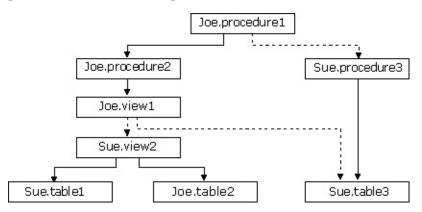
When a user accesses a view, Microsoft® SQL Server<sup>™</sup> does not check permissions on any of the view's underlying objects if these objects and the view are all owned by the same user, and if the view and all its underlying objects are in the same database. If the same user owns a stored procedure and all the views or tables it references, and if the procedure and objects are all in the same database, SQL Server checks only the permissions on the procedure.

If the ownership chain of a procedure or view is broken (not all the objects in the chain are owned by the same user), SQL Server checks permissions on each object in the chain whose next lower link is owned by a different user. In this way, SQL Server allows the owner of the original data to retain control over its accessibility.

Usually, a user who creates a view has to grant permissions only on that view. For example, **Mary** has created a view called **auview1** on the **authors** table, which she also owns. If **Mary** grants **Sue** permission to use **auview1**, SQL Server allows **Sue** access to it without checking permissions on **authors**.

A user who creates a view or stored procedure that depends on an object owned by another user must be aware that any permissions he or she grants depend on the permissions allowed by the other owner.

For example, **Joe** creates a procedure called **procedure1**, which depends on **procedure2** (also owned by **Joe**), and **procedure3** (owned by **Sue**). These procedures in turn depend on other tables and views owned by **Joe** and **Sue**.



**Joe** grants **Mary** permission to use **procedure1**. SQL Server checks the permissions on **procedure1**, **procedure3**, **view2**, **table2**, and **table3** to check that **Mary** is allowed to use them.

# **Using Views as Security Mechanisms**

Views can serve as security mechanisms by restricting the data available to users. Some data can be accessible to users for query and modification, while the rest of the table or database is invisible and inaccessible. Permission to access the subset of data in a view must be granted, denied, or revoked, regardless of the set of permissions in force on the underlying table(s).

For example, the **salary** column in a table contains confidential employee information, but the rest of the columns contain information that should be available to all users. You can define a view that includes all of the columns in the table with the exception of the sensitive **salary** column. As long as table and view have the same owner, granting SELECT permissions on the view allows the user to see nonconfidential columns in the view without having any permissions on the table itself.

By defining different views and granting permissions selectively on them, users, groups, or roles can be restricted to different subsets of data. For example:

- Access can be restricted to a subset of the rows of a base table. For example, define a view that contains only rows for business and psychology books and keep information about other types of books hidden from users.
- Access can be restricted to a subset of the columns of a base table. For example, define a view that contains all the rows of the **titles** table but omits the **royalty** and **advance** columns because this information is sensitive.
- Access can be restricted to a row-and-column subset of a base table.
- Access can be restricted to the rows that qualify for a join of more than one base table. For example, define a view that joins the **titles**, **authors**, and **titleauthor** tables to display the names of authors and books they

have written. This view hides personal data about the authors, and financial information about the books.

- Access can be restricted to a statistical summary of data in a base table. For example, define a view that contains only the average price of each type of book.
- Access can be restricted to a subset of another view or of some combination of views and base tables.

### Permissions and ALTER VIEW

Use the ALTER VIEW Transact-SQL statement to change the definition of a view without having to drop the view and reapply permissions. Any permissions applied to a column in the view are based on the column name defined in the view, rather than the underlying column in the table. Therefore, changing the definition of the view with ALTER VIEW by using the same column name but a different underlying column in a table results in the same permissions for the new column. This example assumes the user **Fred** exists in the **pubs** database:

```
USE pubs
GO
CREATE VIEW v1 AS SELECT title_id, title FROM titles
GO
GRANT SELECT(title_id) ON v1 TO Fred
GO
ALTER VIEW v1 AS SELECT qty AS 'title_id' FROM sales
GO
```

Although the view is altered so that the **title\_id** column name refers to the **qty** column in the **sales** table, rather than the **title\_id** column in the **titles** table, the SELECT permissions granted to **Fred** on the **title\_id** column name still apply.

### See Also

ALTER VIEW
<br/>
CREATE VIEW

# **Using Stored Procedures as Security Mechanisms**

Stored procedures, commonly used as an interface to perform complex activities, can be used to customize security permissions in much the same way as views.

For example, in an archiving scenario, stored procedures can copy data older than a specified interval into an archive table and then delete it from the primary table. Permissions can be used to prevent users from deleting the 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 procedure to ensure that both of these activities are performed together, and then grant users permissions to execute the procedure.

### See Also

CREATE PROCEDURE

# **Advanced Security Topics**

The security topics presented here go beyond the basic use of security in Microsoft<sup>®</sup> SQL Server<sup>™</sup> and provide more detail for specialized applications.

# **Establishing Application Security and Application Roles**

The security system in Microsoft® SQL Server<sup>™</sup> is implemented at the lowest level: the database itself. This is the best method for controlling user activities regardless of the application used to communicate with SQL Server. However, sometimes security controls must be customized to accommodate the special requirements of an individual application, especially when dealing with complex databases and databases with large tables.

Additionally, you may want users to be restricted to accessing data only through a specific application (for example using SQL Query Analyzer or Microsoft Excel) or to be prevented from accessing data directly. Restricting user access in this way prohibits users from connecting to an instance of SQL Server using an application such as SQL Query Analyzer and executing a poorly written query, which can negatively affect the performance of the whole server.

SQL Server accommodates these needs through the use of application roles. Application roles are different than standard roles in that:

• Application roles contain no members.

Microsoft Windows NT® 4.0 or Windows® 2000 groups, users, and roles cannot be added to application roles; the permissions of the application role are gained when the application role is activated for the user's connection through a specific application or applications. A user's association with an application role is due to his ability to run an application that activates the role, rather than his being a member of the role.

- Application roles are inactive by default and require a password to be activated.
- Application roles bypass standard permissions.

When an application role is activated for a connection by the application, the connection permanently loses all permissions applied to the login, user account, or other groups or database roles in all databases

for the duration of the connection. The connection gains the permissions associated with the application role for the database in which the application role exists. Because application roles are applicable only to the database in which they exist, the connection can gain access to another database only through permissions granted to the **guest** user account in the other database. Therefore, if the **guest** user account does not exist in a database, the connection cannot gain access to that database. If the guest user account does exist in the database but permissions to access an object are not explicitly granted to **guest**, the connection cannot access that object, regardless of who created the object. The permissions the user gained from the application role remain in effect until the connection logs out of an instance of SQL Server.

To ensure that all the functions of the application can be performed, a connection must lose default permissions applied to the login and user account or other groups or database roles in all databases for the duration of the connection and gain the permissions associated with the application role. For example, if a user is usually denied access to a table that the application must access, then the denied access should be revoked so the user can use the application successfully. Application roles overcome any conflicts with user's default permissions by temporarily suspending the user's default permissions and assigning them only the permissions of the application role.

Application roles allow the application, rather than SQL Server, to take over the responsibility of user authentication. However, because SQL Server still must authenticate the application when it accesses databases, the application must provide a password because there is no other way to authenticate an application.

If ad hoc access to a database is not required, users and Windows NT 4.0 or Windows 2000 groups do not need to be granted any permissions because all permissions can be assigned by the applications they use to access the database. In such an environment, standardizing on one system-wide password assigned to an application role is possible, assuming access to the applications is secure.

There are several options for managing application role passwords without hardcoding them into applications. For example, an encrypted key stored in the registry (or a SQL Server database), for which only the application has the decryption code, can be used. The application reads the key, decrypts it, and uses the value to set the application role. Using the Multiprotocol Net-Library, the network packet containing the password can also be encrypted. Additionally, the password can be encrypted, before being sent to an instance of SQL Server, when the role is activated.

When an application user connects to an instance of SQL Server using Windows Authentication Mode, an application role can be used to set the permissions the Windows NT 4.0 or Windows 2000 user has in a database when using the application. This method allows Windows NT 4.0 or Windows 2000 auditing of the user account and control over user permissions, while she uses the application, to be easily maintained.

If SQL Server Authentication is used and auditing user access in the database is not required, it can be easier for the application to connect to an instance of SQL Server using a predefined SQL Server login. For example, an order entry application authenticates users running the application itself, and then connects to an instance of SQL Server using the same **OrderEntry** login. All connections use the same login, and relevant permissions are granted to this login.

**Note** Application roles work with both authentication modes.

### Example

As an example of application role usage, a user **Sue** runs a sales application that requires SELECT, UPDATE, and INSERT permissions on the **Products** and **Orders** tables in database **Sales** to work, but she should not have any SELECT, INSERT, or UPDATE permissions when accessing the **Products** or **Orders** tables using SQL Query Analyzer or any other tool. To ensure this, create one user-database role that denies SELECT, INSERT, or UPDATE permissions on the **Products** and **Orders** tables, and add **Sue** as a member of that database role. Then create an application role in the **Sales** database with SELECT, INSERT, and UPDATE permissions on the **Products** and **Orders** tables. When the application runs, it provides the password to activate the application role by using **sp\_setapprole**, and gains the permissions to access the **Products** and **Orders** tables. If **Sue** tries to log in to an instance of SQL Server using any tool except the application, she will not be able to access the **Products** or **Orders** tables.

## To create an application role

⊞ <u>Transact-SQL</u>

# Allowing Other Accounts to Grant Object Permissions

When you grant an object permission to a user account in a database, you can optionally specify the WITH GRANT OPTION clause, which allows the user account to grant that object permission to others. A user account can be a Microsoft® Windows NT® 4.0 or Windows® 2000 user or group or a Microsoft SQL Server<sup>TM</sup> user or role.

For example, if you use the WITH GRANT OPTION clause when you grant permissions on the **salaries** table to the user **user\_a**, **user\_a** is able to grant the same permissions on the table to any other user account in the database. For groups and roles, if you grant permissions on the **salaries** table to role **role\_a** specifying the WITH GRANT OPTION clause, each member of **role\_a** can grant the object permission to any other user account, provided that the AS clause of the GRANT statement is specified. For more information, see <u>GRANT</u>.

**IMPORTANT** When you use the WITH GRANT OPTION clause, you have no future control over which security accounts will receive that permission.

When you revoke a permission granted using the WITH GRANT OPTION clause, specify the CASCADE clause to have the permissions revoked from the user account as well as any other accounts that received the permission from the initial account.

For example, you have granted a permission specifying WITH GRANT OPTION to the user **user\_a**. **User\_a** granted the permission specifying WITH GRANT OPTION to the user **user\_b**, and **user\_b** granted the permission to the user **user\_c**. **User\_a** has left the company, but SQL Server does not allow you to remove a user account if it has granted a permission specifying the WITH GRANT OPTION clause to another account. Specifying the WITH GRANT OPTION clause has created a chain from **user\_a** through **user\_b** to **user\_c**. You cannot remove the account for **user\_a** until the permissions are revoked for **user\_b** and **user\_c**. When you revoke the permission from **user\_a** and specify the CASCADE option, the permission is removed from the **user\_a**, **user\_b**, and **user\_c** accounts. You then may remove the **user\_a** account.

## **Creating SQL Server File Permissions**

Microsoft® SQL Server<sup>™</sup> must create and access files in order to store databases, database backups, error logs, and so on. This SQL Server process must run in the context of a security account with the necessary permissions to create and access these files, whether these files exist on the local computer or a network drive on a remote computer. The security account SQL Server uses depends on the method used to start the instance of SQL Server. If an instance of SQL Server is started:

- As a service on Microsoft Windows NT® 4.0 or Windows® 2000 using the Service Control Manager, SQL Server uses the security account assigned to the SQL Server service.
- At the command prompt, independent of the Service Control Manager, SQL Server uses the security account of the logged on user.
- In Microsoft Windows 98 and Microsoft Windows Millennium Edition, SQL Server uses the security account of the logged on user.

The security account used by SQL Server requires full access permissions to the file system to create, read, write, delete, and execute files. For example, using the NTFS file system, the security account used by SQL Server requires authority to create files with NTFS Full Control permission.

To prevent unauthorized access to the files used by SQL Server, adjust the permissions on the files directly to allow only the security account used by SQL Server access to the files.

**Note** If SQL Server uses the Windows NT 4.0 and Windows 2000 **LocalSystem** built-in security account, file permissions must be granted to the **SYSTEM** account of the local computer running an instance of SQL Server.

#### Securing the Windows NT Registry

SQL Server Setup removes write permissions from the

HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\MSSQLServer\Providers key in the Windows 2000 registry for users who are not SQL Server system administrators. This prevents nonadministrator users from setting the provider options for linked server definitions when using SQL Server Enterprise Manager.

### See Also

Setting up Windows Services Accounts
Starting SQL Server

# **Using Encryption Methods**

Encryption is a method for keeping sensitive information confidential by changing data into an unreadable form. Encryption ensures that data remains secure by keeping the information hidden from everyone, even if the encrypted data is viewed directly. Decryption is the process of changing encrypted data back into its original form so it can be viewed by authorized users.

Microsoft® SQL Server<sup>TM</sup> encrypts or can encrypt:

- Login and application role passwords stored in SQL Server.
- Any data sent between the client and the server as network packets.
- Stored procedure definitions.
- User-defined function definitions.
- View definitions.
- Trigger definitions.
- Default definitions.
- Rule definitions.

**Note** If you are running Microsoft Windows® 2000 and want to use the Windows 2000 Encrypted File System to encrypt any SQL Server files, you must unencrypt the files before you can change the SQL Server service accounts. If you do not unencrypt the files and then reset the SQL Server service accounts, you cannot unencrypt the files.

### Login and Application Role Passwords

Login and application role passwords stored in the SQL Server system tables are always encrypted. This prevents users, including system administrators, from viewing any passwords, including their own. Additionally, application role passwords can be encrypted when the application role is activated before they are sent over the network.

**Note** Using the **sp\_addlogin** system stored procedure, SQL Server logins can be added without encrypting the password, if required. However, this is not recommended unless the passwords are already encrypted because they are being imported from another instance of SQL Server.

## Data in Network Packets

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 an instance of SQL Server). SQL Server can use the Secure Sockets Layer (SSL) to encrypt all data transmitted between an application computer and an instance of SQL Server. The SSL encryption is performed within the Super Socket Net-Library (Dbnetlib.dll and Ssnetlib.dll) and applies to all inter-computer protocols supported by SQL Server 2000. Enabling encryption slows the performance of the Net-Libraries. Encryption forces the following actions in addition to all of the work for an unencrypted connection:

- An extra network round trip is required at connect time.
- All packets sent from the application to the instance of SQL Server must be encrypted by the client Net-Library and decrypted by the server Net-Library.
- All packets sent from the instance of SQL Server to the application must be encrypted by the server Net-Library and decrypted by the client Net-Library.

Shared memory Net-Library communications are inherently secure without the

need for encryption. The shared memory Net-Library does not participates in inter-computer communications. The area of memory shared between the application process and the database engine process cannot be accessed from any other Windows process.

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 is implemented by calling the Windows RPC encryption API. It does not require the use of certificates. The level of RPC encryption, 40-bit or 128-bit, depends on the version of the Windows operating system that is running on the application and database computers. The Multiprotocol Net-Library is not supported by named instances. For more information about SSL, see <u>Net-Library Encryption</u>.

### **Configuring a Multiprotocol Alias**

When you configure a multiprotocol alias, enable encryption. This encryption feature applies only to the Multiprotocol Net-Library. This encryption feature is offered only for compatibility with existing applications. SQL Server clients should use the SSL encryption specified on the **General** tab in the **Enable protocol encryption** check box of the Client Network Utility. For more information on the Client Network Utility, see <u>Configuring Client Net-Libraries</u>.

#### To start the Client Network Utility

## **Revealing SQL Server on a Network**

When you install Microsoft® SQL Server<sup>TM</sup>, SQL Server Setup makes an entry in the Microsoft Windows® 2000 registry that enables Named Pipes clients to see SQL Server in a server enumeration box in SQL Query Analyzer. SQL Server automatically announces itself as a service over Named Pipes to make it easier to locate servers running an instance of SQL Server. However, if you are using Active Directory<sup>TM</sup>, the directory service included in Windows 2000, this functionality is no longer necessary.

Stop SQL Server from announcing itself over Named Pipes by running the NET CONFIG SERVER command with the switch as /HIDDEN:YES. You can reveal the server at any time.

To reveal or cancel the announcement of SQL Server on a network

## **Scripting Data Access Controls in Internet Explorer**

Microsoft<sup>®</sup> SQL Server<sup>™</sup> ships with several data access controls:

- SQL Namespace (SQL-NS)
- SQL Distribution control (replication)
- SQL Merge control (replication)

These controls are signed and marked "safe for initialization and scripting" and can be used in Microsoft Internet Explorer 5 or later.

Before deploying controls that can connect to data sources, you should thoroughly understand the security implications. When you use any of the SQL Server controls, the primary security concern is the ability to run under the authorized user's account through a Windows Authentication login to an instance of SQL Server. A Web page with a scripted control runs with the network identity of the user browsing the page. If the data source connection is based on the connected user's network identity (using Windows Authentication login), the control can access any data that the user browsing the page can access. If a Web page using the control is sent to a user, the control has the permissions of the user browsing the Web page. The control can then read or make changes to databases without the user's knowledge.

To prevent unauthorized access or changes to a database, all the data access controls that are marked as "safe for scripting" take into account security zones settings when being loaded in Internet Explorer version 4.0 or later. If a control is not marked safe for scripting, it can run a script inside of Internet Explorer only at the **Low** security mode of Internet Explorer, and even then only after the user responded to a message stating that a script will be run. Another way to deal with the issue is to remove the user's ability to use a Windows Authenticated login.

Internet Explorer 4.0 does not provide an explicit security option for data access. Therefore, all the controls marked safe for scripting allow, prompt, or disallow scripting based on the security zone being used. The following table shows the

Internet Explorer 4.0 settings.

Security zone	Internet Explorer 4.0 notification
Local computer zone	Controls can be initialized or scripted regardless
	of data source or scripts.
Local intranet zone	User is warned of potential safety violation prior
	to loading the page. User can accept or reject
	initialization or scripting.
Trusted sites zone	Controls can be initialized or scripted regardless
	of data source or scripts.
Internet zone	User is warned of potential safety violation prior
	to loading the page. User can accept or reject
	initialization or scripting.
Restricted sites zone	Scripting errors occur if user attempts to view
	page and execute script.

In contrast to Internet Explorer 4.0, Internet Explorer 5 supports an explicit security option for data access called "Access data sources across domains." This option can be customized, and the setting of this action is used to determine how the controls behave when they are run in Internet Explorer 5. The default settings in Internet Explorer 5 are the same as the programmed settings in Internet Explorer 4.0.

As with all security concerns, you must take specific actions to safeguard your system. SQL Server is protected from security problems only if users with the ability to use Windows Authenticated logins configure the security settings correctly, and answer all security prompts correctly.

**Note** These general steps to safeguard your system apply to any scripting host, including Microsoft Excel spreadsheets or Microsoft Word documents. Users who have the ability to use Windows Authenticated logins should always enable the macro warning feature or similar security setting of an application to detect and prevent any attacks on data.

See Also

Developing SQL-DMO Applications
Programming SQL-NS Applications

# **Auditing SQL Server Activity**

Microsoft® SQL Server<sup>™</sup> 2000 provides auditing as a way to trace and record activity that has happened on each instance of SQL Server (for example, successful and failed logins). SQL Server 2000 also provides an interface, SQL Profiler, for managing audit records. Auditing can only be enabled or modified by members of the **sysadmin** fixed security role, and every modification of an audit is an auditable event.

There are two type of auditing:

- Auditing, which provides some level of auditing but does not require the same number of policies as C2 auditing.
- C2 auditing, which requires that you follow very specific security policies. For more information about C2 auditing, see <u>C2 Auditing</u>.

Both types of auditing can be done by using SQL Profiler.

### **Using SQL Profiler**

SQL Profiler provides the user interface for auditing events. There are several categories of events that can be audited using SQL Profiler, such as:

- End user activity (all SQL commands, logout/login, enabling of application roles).
- DBA activity (DDL, other than grant/revoke/deny and security events, Configuration (DB or server).
- Security events (grant/revoke/deny, login user/role add/remove/configure).
- Utility events (backup/restore/bulk insert/BCP/DBCC commands.

- Server events (shutdown, pause, start).
- Audit events (add audit, modify audit, stop audit).

For more information about what categories of events can be monitored, see <u>Security Audit Event Category</u>.

It is possible to audit the following aspects of SQL Server through SQL Profiler:

- Date and time of event.
- User who caused the event to occur.
- Type of event.
- Success or failure of the event.
- The origin of the request (for example, the Microsoft Windows NT® 4.0 computer name).
- The name of the object accessed.
- Text of the SQL statement (passwords replaced with \*\*\*\*).
- If you are a member of the **sysadmin** or **securityadmin** fixed server role and you reset your own password by using **sp\_password** with all three arguments specified ('*old\_password*', '*new\_password*', '*login*'), the audit record will reflect that you are changing someone else's password.

Auditing can have a significant performance impact. If all audit counters are turned on for all objects, the performance impact could be high. It is necessary to evaluate how many events need to be audited compared to the resulting performance impact. Audit trail analysis can be costly, so it is recommended that audit activity be run on a server separate from the production server. **Note** If SQL Server is started with the **-f** flag, auditing will not run.

### See Also

Monitoring with SQL Profiler

# **Using Audit Logs**

SQL Profiler system stored procedures support file rollover. The maximum file size for the audit log is fixed at 200 megabytes (MB). When the audit log file reaches 200 MB, a new file will be created and the old file handle will be closed. If the directory fills up (for example, if the disk quota for the user of the service account has filled up or the disk is full), then the instance of Microsoft® SQL Server<sup>™</sup> is stopped. The system administrator needs to either free up disk space for the audit log before restarting the instance of SQL Server or restart the instance of SQL Server (if auditing is not configured to start automatically).

Use file rollover to prevent the audit trace from failing because the audit log filled up. However, SQL Server will not shut down unless the user specifically requested this feature when they created the trace. An audit failure produces an entry in the Microsoft Windows® event log and the SQL Server error log.

It is strongly recommended that during SQL Server Setup you create a new directory to contain your audit files. \mssql\audit is the suggested path. If you are running SQL Server on a named instance, the suggested path is MSSQL\$Instance\audit.

# **C2** Auditing

C2 auditing is necessary if you are running a C2 certified system. A C2 certified system meets a government standard that defines the security level. To have a C2 certified Microsoft® SQL Server<sup>™</sup>, you must configure SQL Server in the evaluated C2 configuration. For more information about C2 certification, see the C2 Administrator's and User's Security Guide.

# **Managing Security**

To ensure that data and objects stored in Microsoft® SQL Server<sup>™</sup> are accessed only by authorized users, security must be set up correctly. Security elements that may have to be set up include authentication modes, logins, users, roles, granting, revoking, and denying permissions on Transact-SQL statements and objects, and data encryption.

## How to set up Windows Authentication Mode security (Enterprise Manager)

#### To set up Windows Authentication Mode security

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. On the **Security** tab, under **Authentication**, click **Windows only**.
- 4. Under **Audit level**, select the level at which user accesses to Microsoft® SQL Server<sup>™</sup> are recorded in the SQL Server error log:
  - None causes no auditing to be performed.
  - **Success** causes only successful login attempts to be audited.
  - **Failure** causes only failed login attempts to be audited.
  - All causes successful and failed login attempts to be audited.

#### See Also

Authentication Modes

# How to set up Mixed Mode security (Enterprise Manager)

#### To set up Mixed Mode security

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Security** tab.
- 4. Under Authentication, click SQL Server and Windows.
- 5. Under **Audit level**, select the level at which user accesses to Microsoft® SQL Server<sup>™</sup> are recorded in the SQL Server error log:
  - None causes no auditing to be performed.
  - **Success** causes only successful login attempts to be audited.
  - **Failure** causes only failed login attempts to be audited.
  - All causes successful and failed login attempts to be audited.

#### See Also

Authentication Modes

# How to grant a Windows user or group login access to SQL Server (Enterprise Manager)

To grant a Windows NT 4.0 or Window 2000 user or group login access to SQL Server

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, right-click **Logins**, and then click **New Login**.
- 3. In the **Name** box, enter the Microsoft® Windows NT® 4.0 or Windows® 2000 account (in the form DOMAIN\User) to be granted access to Microsoft SQL Server<sup>™</sup>.
- 4. Under Authentication, click Windows Authentication.
- 5. Optionally:
  - In **Database**, click the default database to which the user is connected after logging into an instance of SQL Server.
  - In **Language**, click the default language in which messages are displayed to the user.

#### See Also

Adding a Windows NT User or Group

# How to grant a Windows user or group access to a database (Enterprise Manager)

To grant a Windows NT 4.0 or Windows 2000 user or group access to a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the user or group will be granted access.
- 3. Right-click **Users**, and then click **New Database User**.
- In the Login name box, type or select the Microsoft® Windows NT® 4.0 or Windows® 2000 user or group name to which database access will be granted.
- 5. Optionally, in **User name**, enter the user name that the login is known by in the database. By default, it is set to the login name.
- 6. Optionally, select database role memberships to be granted to the user or group in addition to **public**, the default.

#### See Also

Granting a Windows NT User or Group Access to a Database

## How to add a SQL Server login (Enterprise Manager)

#### To add a SQL Server login

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, right-click **Logins**, and then click **New Login**.
- 3. In **Name**, enter a name for the Microsoft<sup>®</sup> SQL Server<sup>™</sup> login.
- 4. Under Authentication, select SQL Server Authentication.
- 5. Optionally, in **Password**, enter a password.
- 6. Optionally:
  - In **Database**, click the default database to which the login is connected after logging into an instance of SQL Server.
  - In **Language**, click the default language in which messages are displayed to the user.

#### See Also

Adding a SQL Server Login

# How to add a linked server login (Enterprise Manager)

#### To add a linked server login

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Linked Servers**.
- 3. In the details pane, right-click the linked server to which the login will be added, and then click **Properties**.
- 4. On the **Security** tab, click the local login to add.
- 5. Optionally, select the **Impersonate** check box if the local login should connect to the linked server using its own user security credentials.
- 6. Enter the remote user and remote password with which the local login should connect to the linked server when not using the user's security credentials (**Impersonate** not selected).

## See Also

Establishing Security for Linked Servers

How to set up a linked server (Enterprise Manager)

# How to grant a SQL Server login access to a database (Enterprise Manager)

#### To grant a SQL Server login access to a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the login will be granted access.
- 3. Right-click **Users**, and then click **New Database User**.
- 4. In the **Login name** box, click the Microsoft® SQL Server<sup>™</sup> login to which database access will be granted.
- 5. Optionally, in **User name**, enter the user name that the login is known by in the database. By default, it is set to the login name.
- 6. Optionally, select database role memberships in addition to **public**, the default.

### See Also

Granting a SQL Server Login Access to a Database

<u>guest User</u>

# How to create a SQL Server database role (Enterprise Manager)

#### To create a SQL Server database role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database in which to create a role.
- 3. Right-click **Roles**, and then click **New Database Role**.
- 4. In the **Name** box, enter the name of the new role.
- 5. Optionally, click **Add** to add members to the **Standard role** list, and then click a user or users to add.

Only users in the selected database can be added to the role.

### See Also

Creating User-Defined SQL Server Database Roles

# How to add a member to a SQL Server database role (Enterprise Manager)

#### To add a member to a SQL Server database role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database in which the role exists.
- 3. Click **Roles**.
- 4. In the details pane, right-click the role to which the user will be added, and then click **Properties**.
- 5. Click **Add**, and then click a user or users to add.

Only users in the selected database can be added to the role.

## See Also

Adding a Member to a Predefined Role

Adding a Member to a SQL Server Database Role

# How to add a member to a fixed server role (Enterprise Manager)

#### To add a member to a fixed server role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Server Roles**.
- 3. In the details pane, right-click the role, and then click **Properties**.
- 4. On the **General** tab, click **Add**, and then click the logins to add.

### See Also

Adding a Member to a Predefined Role

# How to grant SQL Server login access to a user by using the Create Login Wizard (Enterprise Manager)

To grant SQL Server login access to a user by using the Create SQL Server Login Wizard

- 1. On the **Tools** menu, click **Wizards**.
- 2. In the **Select Wizard** dialog box, expand **Database**, and then doubleclick **Create Login Wizard**.
- 3. Complete the steps in the wizard.

#### See Also

Using the Create Login Wizard

# How to view a SQL Server login or Windows user or group (Enterprise Manager)

To view a SQL Server login or Windows NT 4.0 or Windows 2000 user or group

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Logins**.
- 3. In the details pane, right-click the login to view, and then click **Properties**.

### See Also

**Viewing Logins** 

## How to view a database user (Enterprise Manager)

#### To view a database user

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the user belongs.
- 3. Click Users.
- 4. In the details pane, right-click the user to view, and then click **Properties**.

### See Also

Viewing Database Users

# How to change the password of a SQL Server login (Enterprise Manager)

#### To change the password of a SQL Server login

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Logins**.
- 3. In the details pane, right-click the login to modify, and then click **Properties**.
- 4. In the **Password** box, on the **General** tab, enter a new password.
- 5. Confirm the password.

### See Also

**Modifying Logins** 

# How to change the default database of a login (Enterprise Manager)

#### To change the default database of a login

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Logins**.
- 3. In the details pane, right-click the login to modify, and then click **Properties**.
- 4. In the **Database** list, on the **General** tab, click the new default database to which the login is connected after logging into an instance of Microsoft® SQL Server<sup>TM</sup>.

## See Also

**Modifying Logins** 

# How to change the default language of a login (Enterprise Manager)

#### To change the default language of a login

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Logins**.
- 3. In the details pane, right-click the login to modify, and then click **Properties**.
- 4. In the **Language** list, on the **General** tab, click the new default language in which messages are to be displayed to the user.

### See Also

**Modifying Logins** 

# How to remove a user or group from a database (Enterprise Manager)

#### To remove a user or group from a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the user or group belongs.
- 3. Click Users.
- 4. In the details pane, right-click the user or group to remove, and then click **Delete**.
- 5. Confirm the deletion.

## See Also

# How to remove a SQL Server login (Enterprise Manager)

#### To remove a SQL Server login

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Logins**.
- 3. In the details pane, right-click the login to remove, and then click **Delete**.
- 4. Confirm the deletion.

## See Also

# How to revoke a Windows user or group login access from SQL Server (Enterprise Manager)

To revoke a Windows NT 4.0 or Windows 2000 user or group login access from SQL Server

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Logins**.
- 3. In the details pane, right-click the Microsoft® Windows NT® 4.0 or Windows® 2000 user or group to revoke, and then click **Delete**.
- 4. Confirm the deletion.

#### See Also

# How to deny login access to a Windows user or group (Enterprise Manager)

To deny login access to a Windows NT 4.0 or Windows 2000 user or group

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Logins**.
- 3. In the details pane, right-click the Microsoft® Windows NT® 4.0 or Windows® 2000 user or group to deny, and then click **Properties**.
- 4. Under **Authentication**, click **Deny access**.

### See Also

Denying Login Access to Windows NT Accounts

<u>How to grant a Windows NT user or group login access to SQL Server</u> (Enterprise Manager)

# How to remove a linked server login (Enterprise Manager)

### To remove a linked server login

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Linked Servers**.
- 3. In the details pane, right-click the linked server to which the linked server login to be removed is mapped, and then click **Properties**.
- 4. On the **Security** tab, under **Local login**, click the linked server login to remove, and then select the blank login at the top of the list.

## See Also

# How to view the roles defined in the current database (Enterprise Manager)

To view the roles defined in the current database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to view.
- 3. Click **Roles**.

## See Also

**Viewing Roles** 

# How to view the fixed server roles (Enterprise Manager)

### To view the fixed server roles

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Server Roles**.

## See Also

Viewing Roles

# How to view the members of a database role (Enterprise Manager)

#### To view the members of a database role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the user belongs.
- 3. Click **Roles**.
- 4. In the details pane, right-click the role to view, and then click **Properties** to view members.

### See Also

# How to remove a user account from a database role (Enterprise Manager)

## To remove a user account from a database role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database in which the role exists.
- 3. Click **Roles**.
- 4. In the details pane, right-click the role to which the user account belongs, and then click **Properties**.
- 5. Select the user to remove, and then click **Remove**.

## See Also

# How to view the members of a fixed server role (Enterprise Manager)

To view the members of a fixed server role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Server Roles**.
- 3. In the details pane, right-click the server role to view, and then click **Properties**.

## See Also

# How to remove a login from a fixed server role (Enterprise Manager)

#### To remove a login from a fixed server role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Server Roles**.
- 3. In the details pane, right-click the server role to modify, and then click **Properties**.
- 4. On the **General** tab, select the login to remove, and then click **Remove**.

## See Also

# How to remove a SQL Server role (Enterprise Manager)

#### To remove a SQL Server role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database in which the role exists.
- 3. Click **Roles**.
- 4. In the details pane, right-click the role, and then click **Delete**.

**Note** You must drop all role members before you can delete the role. Fixed roles cannot be deleted.

• Confirm the deletion.

## See Also

Removing a SQL Server Database Role

# How to allow access by granting permissions (Enterprise Manager)

#### To allow access by granting permissions (on an object)

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the object belongs.
- 3. Depending on the type of object, click one of the following:
  - Tables
  - Views
  - Stored Procedures
- 4. In the details pane, right-click the object on which to grant permissions, point to **All Tasks**, and then click **Manage Permissions**.
- 5. Click **List all users/user-defined database roles/public**, and then select the permission to grant each user.

A check indicates a granted permission. Only permissions applicable to the object are listed.

## See Also

**Granting Permissions** 

# How to grant statement permissions to users within a database (Enterprise Manager)

To grant statement permissions to users within a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, right-click the database containing the users to whom statement permissions will be granted, and then click **Properties**.
- 3. On the **Permissions** tab, select the statement permission to grant each user.

A check indicates a granted permission.

## See Also

**Granting Permissions** 

# How to grant permissions on multiple objects to a user, group, or role (Enterprise Manager)

#### To grant permissions on multiple objects to a user, group, or role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the user, group, or role belongs.
- 3. Depending on the type of user, group, or role to which permissions will be granted, click either **Users** or **Roles**.
- 4. In the details pane, right-click the user, group, or role to which permissions will be granted, point to **All Tasks**, and then click **Manage Permissions**.
- 5. Click **List all objects**, and then select the permission to grant each object.

A check indicates a granted permission. Only permissions applicable to the object are listed.

### See Also

**Granting Permissions** 

# How to prevent access by denying permissions (Enterprise Manager)

#### To prevent access by denying permissions (on an object)

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the object belongs.
- 3. Depending on the type of object to which access will be denied, click one of the following:
  - Tables
  - Views
  - Stored Procedures
- 4. In the details pane, right-click the object to which access will be denied, point to **All Tasks**, and then click **Manage Permissions**.
- 5. Click **List all users/user-defined database roles/public**, and then select the permission to deny each user.

An 'X' indicates a denied permission. Only permissions applicable to the object are listed.

## See Also

**Denying Permissions** 

## How to deny statement permissions from users within a database (Enterprise Manager)

To deny statement permissions from users within a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, right-click the database containing the users to whom statement permissions will be denied, and then click **Properties**.
- 3. On the **Permissions** tab, select the statement permission to deny each user.

An 'X' indicates a denied permission.

## See Also

**Denying Permissions** 

# How to deny permissions on multiple objects to a user, group, or role (Enterprise Manager)

#### To deny permissions on multiple objects to a user, group, or role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the user, group, or role belongs.
- 3. Depending on the type of user, group, or role to which permissions will be denied, click either **Users** or **Roles**.
- In the details pane, right-click the user or group to which permissions will be denied, point to All Tasks, and then click Manage Permissions. If you are denying permission to a role, right-click the role to which permissions will be denied, click Properties, and then click Permissions.
- 5. Click **List all objects**, and then select the permission to deny for each object.

An 'X' indicates a denied permission. Only permissions applicable to the object are listed.

### See Also

**Denying Permissions** 

# How to revoke permissions on an object (Enterprise Manager)

#### To revoke permissions on an object

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the object belongs.
- 3. Depending on the type of object to which access will be revoked, click one of the following:
  - Tables
  - Views
  - Stored Procedures
- 4. In the details pane, right-click the object to which access will be revoked, point to **All Tasks**, and then click **Manage Permissions**.
- 5. Click **List all users/user-defined database roles/public**, and then select the permission to revoke from each user.

An empty box indicates a revoked permission. Only permissions applicable to the object are listed.

## See Also

**Revoking Permissions** 

# How to revoke statement permissions from users in a database (Enterprise Manager)

To revoke statement permissions from users in a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, right-click the database containing the users from whom statement permissions will be revoked, and then click **Properties**.
- 3. On the **Permissions** tab, select the statement permission to revoke from each user.

An empty box indicates a revoked permission.

## See Also

**Revoking Permissions** 

# How to revoke permissions on multiple objects from a user, group, or role (Enterprise Manager)

#### To revoke permissions on multiple objects from a user, group, or role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the user, group, or role belongs.
- 3. Depending on the type of user, group, or role from which permissions will be revoked, click either **Users** or **Roles**.
- 4. In the details pane, right-click the user or group from which permissions will be revoked, point to All Tasks, and then click Manage Permissions. If you are revoking permission from a role, right-click the role to which permissions will be denied, click Properties, and then click Permissions.
- 5. Click **List all objects**, and then select the permission to revoke for each object.

An empty box indicates a revoked permission. Only permissions applicable to the object are listed.

### See Also

**Revoking Permissions** 

# How to create an application role (Enterprise Manager)

#### To create an application role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database in which to create a role.
- 3. Right-click **Roles**, and then click **New Database Role**.
- 4. In the **Name** box, enter the name of the new application role.
- 5. Under **Database role type,** click **Application role**, and then enter a password.

## See Also

Establishing Application Security and Application Roles

# How to remove an application role (Enterprise Manager)

#### To remove an application role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database in which the application role exists.
- 3. Click **Roles**.
- 4. In the details pane, right-click the application role to remove, and then click **Delete**.
- 5. Confirm the deletion.

## See Also

Establishing Application Security and Application Roles

## How to reveal or cancel announcement of SQL Server on a network (Windows)

To reveal or cancel announcement of SQL Server on a network

- 1. In **Control Panel**, double-click **Network**.
- 2. Click the **Services** tab.
- 3. In the **Network Services** list, click **Server**, and then click **Properties**.
- 4. Select **Make Browser Broadcasts to LAN Manager 2.x Clients** to reveal the server, or clear the check box to hide the server.

### See Also

**Revealing SQL Server on a Network** 

# How to grant, deny, or revoke permissions on multiple objects to a user-defined role (Enterprise Manager)

To grant, deny, or revoke permissions on multiple objects to a user-defined role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the role belongs.
- 3. Click **Roles**.
- 4. In the details pane, right-click the user-defined role to which permissions will be granted, denied, or revoked, and then click **Properties**.
- 5. Under **Names**, click **Permissions**.
- 6. Click **List all objects**, and then select the permission to grant, deny, or revoke on each object.

A checkmark indicates a granted permission; an 'X' indicates a denied permission; and an empty box indicates a revoked permission. Only permissions applicable to the object are listed.

# How to start the default instance of SQL Server (Service Manager)

To start the default instance of SQL Server

1. In the **Services** box, click **SQL Server**.

If the service is a remote service, type the name of the remote server in the **Server** box.

2. Click **Start/Continue**.

### See Also

Starting SQL Server Manually

### How to start a clustered instance of SQL Server (Service Manager)

#### To start a clustered instance of SQL Server

- 1. Type the name of the virtual SQL Server in the **Server** box. If it is a default instance, you only need to specify the virtual server name. If it is a named instance, you must enter VIRTUALSERVER\Instance.
- 2. In the **Services** box, click **SQL Server**.
- 3. Click **Start/Continue**.

#### See Also

Starting SQL Server Manually

# How to start a named instance of SQL Server (Service Manager)

#### To start a named instance of SQL Server

- 1. In the **Server** box, select the name of the server and the named instance of Microsoft® SQL Server<sup>™</sup> 2000, or type the name of the remote server.
- 2. In the **Services** box, click **SQL Server**, and then click **Start/Continue**.

# How to start the default instance of SQL Server (Windows)

To start the default instance of SQL Server

- 1. In **Control Panel**, double-click **Services**.
- 2. In the **Services** dialog box, click **MSSQLSERVER**, and then click **Start**.

# How to start a named instance of SQL Server (Windows)

#### To start a named instance of SQL Server

- 1. In **Control Panel**, double-click **Services**.
- 2. In the **Services** dialog box, click the named instance of Microsoft® SQL Server<sup>™</sup> 2000 you want to start, and then click **Start**.

## How to start the default instance of SQL Server (Command Prompt)

#### To start the default instance of SQL Server from a command prompt

• From a command prompt, enter:

sqlservr.exe -c

**Note** You must switch to the appropriate directory (for the instance of Microsoft® SQL Server<sup>TM</sup> you want to start) in the command window before starting sqlservr.exe.

### See Also

Starting SQL Server Manually

## How to start a named instance of SQL Server (Command Prompt)

#### To start a named instance of SQL Server from a command prompt

• From a command prompt, enter this command:

```
sqlservr.exe -c -s {instancename}
```

**Note** You must switch to the appropriate directory (for the instance of Microsoft® SQL Server<sup>™</sup> 2000 you want to start) in the command window before starting sqlservr.exe. For example, if Instance1 uses \mssql\$Instance1 to store its binaries, you must be in the \mssql\$Instance1\binn directory to start sqlservr.exe.

# How to start the default instance of SQL Server in single-user mode (Command Prompt)

To start the default instance of SQL Server in single-user mode from a command prompt

• From a command prompt, enter:

#### sqlservr.exe -c -m

**Note** You must switch to the appropriate directory (for the instance of Microsoft® SQL Server<sup>™</sup> you want to start) in the command window before starting sqlservr.exe.

#### See Also

Starting SQL Server in Single-User Mode

**Using Startup Options** 

# How to start a named instance of SQL Server in single-user mode (Command Prompt)

To start a named instance of SQL Server in single-user mode from a command prompt

• From a command prompt, enter:

```
sqlservr.exe -c - m -s {instancename}
```

**Note** You must switch to the appropriate directory (for the instance of Microsoft® SQL Server<sup>™</sup> 2000 you want to start) in the command window before starting sqlservr.exe.

# How to start the default instance of SQL Server with minimal configuration (Command Prompt)

To start the default instance of SQL Server with minimal configuration

• From a command prompt, enter the following command to start the default instance of Microsoft® SQL Server<sup>™</sup> as a service:

sqlservr -c -f

**Note** You must switch to the appropriate directory (for the instance of SQL Server you want to start) in the command window before starting sqlservr.exe.

### See Also

Starting SQL Server Manually

# How to start a named instance of SQL Server with minimal configuration (Command Prompt)

To start a named instance of SQL Server with minimal configuration

• From a command prompt, enter the following command to start a named instance of Microsoft® SQL Server<sup>™</sup> 2000 as a service:

sqlservr -c -f -s {instancename}

**Note** You must switch to the appropriate directory (for the instance of SQL Server you want to start) in the command window before starting sqlservr.exe.

# How to pause and resume the default instance of SQL Server (Service Manager)

To pause and resume the default instance of SQL Server

1. In the **Services** box, click **SQL Server**.

If the service is a remote service, type the name of the remote server.

2. Click **Pause**, and then click **Start/Continue**.

See Also

Pausing and Resuming SQL Server

## How to stop a clustered instance of SQL Server (Service Manager)

#### To stop a clustered instance of SQL Server

- 1. Type the name of the virtual Microsoft® SQL Server<sup>™</sup> in the **Server** box. If it is a default instance, you only need to specify the virtual server name. If it is a named instance, you must enter VIRTUALSERVER\Instance.
- 2. In the **Services** box, click **SQL Server**.
- 3. Click **Stop**. This pauses the cluster resource, and then stops the **SQL Server** service, which does not cause a failover of SQL Server.

#### See Also

Stopping SQL Server

# How to pause and resume a named instance of SQL Server (Service Manager)

#### To pause and resume a named instance of SQL Server

- 1. In the **Server** box, select the name of the server and the named instance of Microsoft® SQL Server<sup>™</sup> 2000, or type the name of the remote server.
- 2. In the **Services** box, click **SQL Server**.
- 3. Click **Pause**, and then click **Start/Continue**.

# How to pause and resume the default instance of SQL Server (Windows)

To pause and resume the default instance of SQL Server

- 1. In **Control Panel**, double-click **Services**.
- 2. In the **Services** dialog box, click **MSSQLSERVER**.
- 3. Click **Pause** or **Continue**.

# How to pause and resume a named instance of SQL Server (Windows)

To pause and resume a named instance of SQL Server

- 1. In **Control Panel**, double-click **Services**.
- 2. In the **Services** dialog box, click the named instance of Microsoft® SQL Server<sup>™</sup> 2000 you want to pause.
- 3. Click **Pause** or **Continue**.

# How to pause and resume the default instance of SQL Server (Command Prompt)

#### To pause and resume the default instance of SQL Server

• From a command prompt, enter either:

#### net pause mssqlserver

-or-

#### net continue mssqlserver

An instance of Microsoft<sup>®</sup> SQL Server<sup>™</sup> can be paused or resumed only if it was started as a Microsoft Windows NT<sup>®</sup> 4.0 or Windows<sup>®</sup> 2000 service.

### See Also

Pausing and Resuming SQL Server

### How to pause and resume a named instance of SQL Server (Command Prompt)

To pause and resume a named instance of SQL Server

• From a command prompt, enter either:

net pause mssql\$instancename

-or-

net continue mssql\$instancename

# How to broadcast a shutdown message (Command Prompt)

#### To broadcast a shutdown message

• From a command prompt, enter:

```
net send /users "message"
```

For example:

net send /users "SQL Server is going down in 20 minutes. Disconnect within 15 minutes."

**Note** The shutdown message can be broadcast only if an instance of Microsoft® SQL Server<sup>™</sup> is running on Microsoft Windows NT® 4.0 or Windows® 2000. The **users** option specifies that the message be sent to all users connected to the server. For information about other **net send** options, see the Windows NT 4.0 and Windows 2000 documentation.

#### See Also

Stopping SQL Server

## How to stop the default instance of SQL Server (Windows)

To stop the default instance of SQL Server

- 1. In **Control Panel**, double-click **Services**.
- 2. In the **Services** dialog box, click **MSSQLSERVER**, and then click **Stop**.

### How to stop a named instance of SQL Server (Windows)

#### To stop a named instance of SQL Server

- 1. In **Control Panel**, double-click **Services**.
- 2. In the **Services** dialog box, click the named instance of Microsoft® SQL Server<sup>™</sup> 2000 you want to stop, and then click **Stop**.

## How to stop the default instance of SQL Server (Command Prompt)

#### To stop the default instance of SQL Server

• From a command prompt, enter:

#### net stop mssqlserver

**Note** Stopping a default instance of Microsoft® SQL Server<sup>™</sup> using SQL Server Enterprise Manager or the **net stop mssqlserver** command causes SQL Server to perform a checkpoint in all databases. Then a SHUTDOWN WITH NOWAIT is done to flush all committed data from the data cache and to stop the server immediately. Stopping a default instance of SQL Server from the command prompt works only if you are running Microsoft Windows NT® 4.0 or Windows® 2000.

### See Also

Stopping SQL Server

## How to stop a named instance of SQL Server (Command Prompt)

#### To stop a named instance of SQL Server

• From a command prompt, enter:

#### net stop mssql\$instancename

**Note** Stopping a named instance of Microsoft® SQL Server<sup>™</sup> 2000 using SQL Server Enterprise Manager or the **net stop mssql\$instancename** command causes SQL Server to perform a checkpoint in all databases. Then a SHUTDOWN WITH NOWAIT is done to flush all committed data from the data cache and to stop the server immediately. Stopping a named instance of SQL Server 2000 from the command prompt works only if you are running Microsoft Windows NT® 4.0 or Windows® 2000.

## How to log in to the default instance of SQL Server (Command Prompt)

To log in to the default instance of SQL Server

From a command prompt, enter either:
 osql /U [login\_id] /P [password] /S [servername]
 -or-

isql/U [login\_id]/P [password] /S [servername]

### See Also

osql Utility

## How to log in to a named instance of SQL Server (Command Prompt)

To log in to a named instance of SQL Server

• From a command prompt, enter either:

osql / U login\_id /P password /S servername\instancename

-or-

isql/U login\_id/P password /S servername\instancename

### How to change the default service (Service Manager)

#### To change the default service

- 1. Right-click SQL Server Service Manager, and then click **Options**.
- 2. In the **Default Service** box, select the new default service to view through SQL Server Service Manager. When you restart the computer, the service that appears is the new default. For example, if you change the default service to SQLServerAgent service and then shut down the computer, the next time you start it, SQLServerAgent service will be displayed in Service Control Manager. You can only change the default service for the local machine.

## **Installing Analysis Services**

This section contains information about installing Microsoft® SQL Server<sup>™</sup> 2000 Analysis Services only. It does not contain information about installing other components of SQL Server 2000. For more information about installing other components, such as English Query, see <u>Getting Started with SQL Server</u> <u>Books Online</u>.

Торіс	Description
Hardware and Software	Provides the hardware and software
Requirements for Installing	requirements for installing and running
Analysis Services	Analysis Services.
Running Setup	Provides step-by-step instructions to
	install Analysis Services.
Setup Parameters and Silent	Describes the parameters for the
<u>Installation</u>	Analysis Services Setup program
	(Setup.exe).
Reinstalling Analysis Services	Describes how to reinstall Analysis
	Services.
Removing Analysis Services	Describes how to remove Analysis
	Services.
Upgrading from an Earlier Version	Describes how to upgrade from an
	earlier version of Analysis Services.
Backward Compatibility	Provides information about
	compatibility with previous versions of
	Analysis Services (formerly called
	OLAP Services).

This section contains the following topics.

### **Related Documents**

The Readme.html file in the root directory of the SQL Server 2000 CD-ROM contains information about Analysis Services. You can also view the release notes by clicking **Read the Release Notes** on the SQL Server 2000 Setup

program (Autorun.exe) menu.

## Hardware and Software Requirements for Installing Analysis Services

Before you can install Microsoft® SQL Server 2000<sup>™</sup> Analysis Services, your computer must meet the following requirements.

Hardware/software	Requirements
Computer	Intel® or compatible (Pentium 133 MHz or higher, Pentium PRO, Pentium II, or Pentium III)
Memory (RAM)	32 megabytes (MB) minimum (64 MB recommended)
Disk drive	CD-ROM drive
Hard disk space <sup>(1)</sup>	50 – 90 MB (130 MB for all components including common files and samples), 12 MB for the client only
Operating system	Microsoft Windows® 2000 Server <sup>(3)</sup> -or-
	Microsoft Windows NT® Server 4.0 with Service Pack 5 or later <sup>(3)</sup>
	For client components on client computers only, the following systems also qualify:
	Windows 2000 Professional Windows NT Workstation 4.0 with Service Pack 5 Windows 98 Windows 95 + DCOM95 Windows 95 OSR2 + DCOM95
Network software	Windows 2000, Windows NT 4.0, Windows 98, or Windows 95 built-in network software; and TCP/IP (included with Windows).
Online product documentation viewer	Microsoft Internet Explorer version 5.0 or later <sup>(2)</sup> . You must install Windows NT 4.0 Service Pack 5 or

Access permissions	To install the services for Analysis server, you must
	be logged on to the server with Administrator
	permissions.

1 Setup installs a number of components that can be shared by other applications and may already exist on the computer.

2 Internet Explorer is required for Microsoft Management Console (MMC) and HTML Help. A minimal installation is sufficient, and Internet Explorer does not need to be your default browser. Internet Explorer is not required for the client-only installation.

3 Analysis Services should not be installed on a domain controller; this installation configuration is not supported.

For more information about supported hardware, see the Microsoft Windows Hardware Compatibility List at the <u>Microsoft Web site</u>. For more information about Windows 2000-compatible hardware, use the Microsoft Windows 2000 compatible hardware devices search tool at the <u>Microsoft Web site</u>.

## **Running Setup**

This topic describes how to install Microsoft® SQL Server<sup>™</sup> 2000 Analysis Services.

If you are upgrading from an earlier version of Analysis Services (formerly called OLAP Services), you should take certain steps before performing the following procedure. For more information, see <u>Upgrading from an Earlier</u> <u>Version</u>.

If you are reinstalling Analysis Services, you should take certain steps before and after performing the following procedure. For more information, see <u>Reinstalling Analysis Services</u>.

Although Analysis Services can connect to multiple instances of SQL Server running on a single computer, you cannot install multiple instances of Analysis Services on a single computer.

To install Analysis Services, use the Analysis Services Setup program or the SQL Server 2000 Setup program.

#### **To install Analysis Services**

- 1. Exit all Microsoft Windows® applications.
- 2. Insert the SQL Server 2000 CD into the CD-ROM drive. This starts the SQL Server 2000 Setup program. If the Setup program does not start automatically, run the Autorun.exe program in the root directory of the CD-ROM.
- 3. Click Install SQL Server 2000 Components.
- 4. Click **Analysis Services** to start the Analysis Services Setup program.
- 5. In the **Welcome** step, click **Next**.

- 6. In the **Software License Agreement** step, read the license agreement, and then do one of the following:
  - To accept the license agreement, click **Yes**. You must select this option to install Analysis Services.
  - To reject the license agreement, click **No**. If you select this option, the program will ask you to confirm exiting. If you select **Exit Setup**, the program closes and the installation is canceled. To continue Setup, click **Resume**.
- 7. The Setup program prompts you to enter the CD key. Type the 10-digit CD key for the product, and then click **OK**.
- 8. The Setup program displays the complete product ID, which you can record for future reference. After you record the product ID, click **OK**.
- 9. In the **Select Components** step, select the components you want to install. All of the options are selected by default. You cannot clear the check box of any component on which another selected component depends.

Unless you are installing the client components on a client computer, installing all components is recommended. The following components are available for installation.

Component	Description
Analysis server	Binary executables and other server-
	related files required for an installation
	of an Analysis server. Includes the
	FoodMart 2000 sample database used
	by the tutorial. Requires the client
	components.
Analysis Manager	Binary executables and other files that
	support the user interface for
	administering the Analysis server.
	Includes the MDXSample executable

	file. Requires Decision Support Objects (DSO) and the client components.
Decision Support	The object model for administering the
Objects	Analysis server and managing meta data. Requires the client components.
Client components	Binary executables and related files for the Analysis Services client. Client components include PivotTable® Service.
Sample applications	Sample applications include the MDXSample source files, the <b>FoodMart</b> <b>2000</b> database, and programming samples. Requires the client components.
Books Online	The entire documentation set for SQL Server 2000, including Analysis Services. This file is approximately 30 megabytes (MB). If space is at a premium, you can choose not to install Books Online. However, product documentation will not be available in the user interface until it is reinstalled.

To change the destination drive or folder, click **Browse**. Although remote network drives are listed in these dialog boxes, installation to locations on remote network drives is not supported.

**Space Required** and **Space Available** indicate disk drive space and help you determine what components to install. If your current disk drive does not have enough space available, you can click **Disk Space** to determine which disks on your computer have enough space to install Analysis Services.

After you select the components to install, click **Next**. The steps that follow may change depending on which components you selected to install.

10. In the **Data Folder Location** step, you can change the location of the

Data folder, which is the data storage location of the Analysis server.

The default location for the Data folder is C:\Program Files\Microsoft Analysis Services\Data (unless you specified another location for Analysis Services in the previous step). You can specify a different location by clicking **Browse**. If you change the default folder or drive, be sure to enter a fully qualified path. To specify a data storage location other than the computer on which the server is installed, you must have full control access permissions on that computer.

**IMPORTANT** The Data folder stores security files that control end users' access to objects on the Analysis server. For this reason, the Data folder must be secured against unauthorized access.

After you select the location of the Data folder, click **Next**.

- 11. In the **Select Program Folder** step, accept the default program folder name or enter a new one. This determines the location of the Analysis Services menu items on the **Start** menu. Click **Next**.
- 12. Analysis Services installation begins. After Setup notifies you that the installation is complete, click **Finish**.
- 13. If you are prompted to restart your computer, do one of the following:
  - Click **Yes, I want to restart my computer now**, and then click **Finish**.
  - Click **No, I will restart my computer later**, and then click **Finish**. If you select this option, the installation is not complete until after you restart the computer.
- 14. If you are finished installing SQL Server 2000 components, click **Exit** in the SQL Server 2000 Setup program.

If in Step 10 you specified a data storage location other than the computer on which the server is installed, you must configure your Analysis server service (MSSQLServerOLAPService) to log on as your user account, instead of the default, which is to log on as the system account. To do this, use the Services application, which is in Control Panel in Windows NT® 4.0 or the Administrative Tools folder in Control Panel in Windows 2000.

### **Setup Parameters and Silent Installation**

You can start the Analysis Services Setup program (\Msolap\Install\Setup.exe on the SQL Server CD-ROM) with the following optional command line parameters:

-r

This option causes Setup.exe to automatically generate a silent response file (.iss), which is a record of the installation input, in the systemroot folder (typically C:\WinNT).

-S

This option performs a silent (unattended) installation.

#### -f1<path\ResponseFile>

This option allows you to specify the alternate location and name of the response file (.iss file). If the -f1 switch is not used when you run silent installation, Setup searches for the response file Setup.iss in the same folder as Setup.exe.

#### -f2<path\LogFile>

This option allows you to specify an alternate location and name of the log file. By default, the Setup.log log file is created and stored in the systemroot folder (typically C:\Winnt).

If you use the **-r** option you can create a record of any installation scenario. You can use this record to perform a silent (unattended) installation. For example, the following command initiates a silent installation of the components specified in the Setup.iss response file previously recorded when you used the **-r** option:

Setup.exe -s -f1C:\temp\setup.iss

-Z

Prevents Setup.exe from checking the available memory during initialization. This switch is necessary when running Setup on a computer with more than 256 megabytes (MB) of memory. If it is not used, Setup.exe reports insufficient memory and exits.

# **Reinstalling Analysis Services**

To reinstall Microsoft® SQL Server<sup>™</sup> 2000 Analysis Services, follow these steps:

- 1. If you have made changes to the FoodMart 2000 sample database and want to preserve changes, back up FoodMart2000.mdb, which is installed by default to: C:\Program Files\Microsoft Analysis Services\Samples. Otherwise, this file is overwritten during the installation process.
- 2. Install Analysis Services. For more information, see <u>Running Setup</u>.

**Note** Reinstalling Analysis Services does not delete the Analysis Services repository (Msmdrep.mdb), which contains Analysis Services meta data. However, you must process all cubes in the repository after reinstallation. If you have backed up the **FoodMart 2000** sample database before reinstallation, restore FoodMart2000.mdb to recover your changes to the file.

### **Stopping or Removing Analysis Services**

To stop Microsoft<sup>®</sup> SQL Server<sup>™</sup> 2000 Analysis Services, follow these steps:

- 1. Open Control Panel.
- 2. If your computer's operating system is Windows® 2000, open the Administrative Tools folder, and then double-click **Services**.

If your computer's operating system is Windows NT® 4.0, doubleclick **Services**.

- 3. Select MSSQLServerOLAPService, and then on the **Action** menu click **Stop**.
- 4. Wait until the application notifies you that the service has stopped.

To remove Analysis Services, use the Add/Remove Programs application in Control Panel. Removing Analysis Services does not delete the Analysis Services repository (Msmdrep.mdb), which contains Analysis Services meta data, or the query log (Msmdqlog.mdb). If you want to fully remove Analysis Services, you must delete these files manually.

## **Upgrading from an Earlier Version**

To upgrade from an earlier version of Microsoft® SQL Server<sup>™</sup> 2000 Analysis Services (previously OLAP Services), perform the following actions:

- Back up the Analysis Services repository and query log.
- Before you install Analysis Services, as a precaution against data loss, back up the Analysis Services repository (Msmdrep.mdb), which contains Analysis Services meta data, and the query log (Msmdqlog.mdb). These files are located in the Bin folder in the Analysis Services folder.Run Setup.

Install Analysis Services by running the Analysis Services Setup program. For more information, see <u>Running Setup</u>.

When you upgrade from an earlier version, Setup does not delete or replace the Analysis Services repository or the query log.

**Note** The default location for Analysis Services has changed from C:\Program Files\OLAP Services in earlier versions of Analysis Services to C:\Program Files\Microsoft Analysis Services in this version of Analysis Services.

# **Backward Compatibility**

Microsoft® SQL Server<sup>™</sup> 2000 Analysis Services is compatible with SQL Server version 7.0 OLAP Services. Cubes that were created in SQL Server 7.0 OLAP Services need to be migrated to the updated meta data repository format and reprocessed. Otherwise, the existing structures for cubes, roles, shared dimensions, and so on do not need to be changed. For more information about migrating the SQL Server 7.0 OLAP Services repository to SQL Server 2000 Meta Data Services, see <u>Migrating Analysis Services Repositories</u>.

The following sections concern backward compatibility with SQL Server 7.0 OLAP Services.

#### **Administration of Analysis Services**

Analysis Manager is backward compatible with SQL Server 7.0 OLAP Services. It is capable of administering both OLAP servers (the server that ships with SQL Server 7.0 OLAP Services), and Analysis servers (the server that ships with SQL Server 2000 Analysis Services) concurrently. When administering an OLAP server, the OLAP Services portion of SQL Server 7.0 Service Pack 2 code is used to assure complete backward compatibility. The add-in programs in Service Pack 2 are now integrated with Analysis Manager and do not need to be installed to administer OLAP servers.

#### **Client and Local Cube Support**

Some features in SQL Server 2000 Analysis Services are not supported by the SQL Server 7.0 OLAP Services client components or in a local cube. For more information, including a list of features, see <u>7.0 Analysis Services Client and Local Cube Support</u>.

#### **Decision Support Objects**

Analysis Services now includes an updated version of Decision Support Objects (DSO), which is automatically installed during Setup. Programs must use this updated version of DSO when administering an Analysis server (the server that ships with SQL Server 2000 Analysis Services). No other change to these

programs is necessary. Programs that use the updated version of DSO are compatible with and can administer OLAP servers (the server that ships with SQL Server 7.0 OLAP Services); however, new features will not be available on the OLAP server.

#### **PivotTable Service**

SQL Server 2000 Analysis Services includes an updated version of PivotTable® Service. Client applications that use PivotTable Service do not need to use this new version when connecting to an Analysis server unless you need access to objects that include new features. The objects that use these new features (such as data mining models and cubes that include parent-child dimensions) are not seen by the client applications that use the earlier version of PivotTable Service. Client applications that use the updated version of PivotTable Service can connect to any server, regardless of its version. Client applications that use the updated version of PivotTable Service can configure their compatibility settings using the following properties:

- MDX Compatibility property
- MDX Unique Name Style property
- Secured Cell Value property
- Visual Mode property

#### **Custom Add-in Programs**

Custom add-in programs that were developed for use with SQL Server 7.0 OLAP Services will continue to work with SQL Server 2000 Analysis Services. No changes are necessary to use them.

### Archiving, Restoring, and Migrating Data

Analysis Services supports some but not all permutations of archiving and restoring databases and migrating repositories between versions of the product.

For information about supported migration paths, see <u>Supported Migration Paths</u> <u>for Analysis Services Repositories</u>. For information about archiving and restoring data between versions of the product, see <u>Archiving and Restoring</u> <u>Databases Between Versions of Analysis Services</u>.

# 7.0 Analysis Services Client and Local Cube Support

This table shows support for new server features by the Microsoft® SQL Server<sup>™</sup> 7.0 OLAP Services client components and in a SQL Server 2000 Analysis Services local cube. When a feature may cause data to be translated incorrectly by a 7.0 client application, the server prevents the cube from being visible and prevents the client connection to the cube. If the absence of a feature in a local cube might change data values presented to the user, then a local cube using the feature cannot be created.

For each feature listed here, the table shows whether a cube containing a feature is visible on a 7.0 client application and if the cube is visible whether the feature itself is available on the 7.0 client application. For each feature, the table also shows whether a local cube can be created using the feature and whether the feature itself is supported in a local cube.

	Cube is	Feature	Can create	Supported
Feature	visible on 7.0 client	available on 7.0 client	local cube using feature	in a local cube
<u>Actions</u>	Yes	No	Yes	No
<u>Additional</u>	Yes	Yes	Yes	(2)
authentication				
<u>methods</u>				
Calculated cells	No	No	No	No
Changing dimensions	Yes	No	Yes	No
Custom member	No	No	No	No
<u>formulas</u>				
Custom rollup	No	No	No	No
<u>formulas</u>				
Default members	No	No	Yes	Yes
Dimension security	No	No	No	No
<u>DistinctCount</u>	No	No	No	No
Drillthrough	Yes	No	Yes	No
Enhanced cell	Yes	Yes	Yes	No
<u>Security</u>				

<u>Enhanced virtual</u> <u>dimensions<sup>(1)</sup></u>	Yes	Yes	Yes	Not applicable
Exceeding 7.0 Limits <sup>(3)</sup>	No	No	Yes	Yes
Linked cubes	Yes	Not applicable	Yes	No
Member groups	Yes	Yes	Yes	Yes
Members with data	Yes	Yes	Yes	Yes
New MDX functions	Yes	No	Yes	(4)
<u>Parent-child</u> <u>dimensions</u>	No	No	Yes	Yes
Ragged dimensions	Yes	Yes	Yes	Yes
ROLAP dimensions	Yes	Not applicable	Yes	No
Siblings with same names	No	No	Yes	Yes
<u>Write-enabled</u> dimensions	Yes	No	Yes	No

1 The earlier limit of 760 members in a virtual dimension does not apply.

2 Cell security is not supported on local cubes.

3 Exceeding 127 measures in a cube, 63 dimensions in a cube, or 128 levels in a cube. For information about SQL Server 2000 Analysis Services limits, see <u>Specifications and Limits</u>.

4 For the SQL Server 7.0 OLAP Services client, new Multidimensional Expressions (MDX) functions are not supported. For local cubes, new MDX functions are available, except for **LookUpCube**. Calculated members using **LookUpCube** in local cubes are not created.

# **Supported Migration Paths for Analysis Services Repositories**

You can migrate a Microsoft® SQL Server<sup>™</sup> 2000 Analysis Services repository from the default Microsoft Access (Microsoft Jet 3.5 or 4.0) database to a SQL Server database on the same or a different computer. You cannot migrate a SQL Server repository to a Microsoft Access repository. You can change the format from native to SQL Server 2000 Meta Data Services format when you migrate a database. To migrate a SQL Server database repository between native and Meta Data Services formats, you must migrate it from one SQL Server database to another. The following table shows supported migration paths for repository databases.

		To native		To MDS
			SQL Server	
		Jet 3.5/4.0	7.0/2000	SQL Server 2000
From native	Jet 3.5/4.0	No	Yes	Yes
	SQL Server 7.0/2000	No	Yes <sup>(2)</sup>	Yes <sup>(2)</sup>
From MDS <sup>(1)</sup>	SQL Server 2000	No	Yes <sup>(2)</sup>	Yes <sup>(2)</sup>

1 MDS represents the Meta Data Services (previously named Microsoft Repository) format supported by SQL Server 2000.

2 Source and destination must be different databases.

#### See Also

Migrating Analysis Services Repositories

OLE DB Provider for Jet

### **Archiving and Restoring Databases Between Versions** of Analysis Services

On an Analysis server (the server that ships with Microsoft SQL Server 2000 Analysis Services), you can restore databases that were archived using an OLAP server (the server that ships with SQL Server 7.0 OLAP Services) or an Analysis server. The following table shows all the restoration paths supported for databases archived while in SQL Server 7.0 OLAP Services or SQL Server 2000 Analysis Services using native or SQL Server 2000 Meta Data Services formats with SQL Server or the Microsoft Jet 3.5 or 4.0 OLE DB provider.

			To native				To MDS
			Jet 3.5/4.0		SQL Server		SQL Server
			7.0	2000	7.0	2000	2000
From native <sup>(1)</sup>	Jet 3.5/4.0	7.0	Yes	Yes	Yes	Yes	Yes
		2000	No(3)	Yes	No(3)	Yes	Yes
	SQL Server	7.0	Yes	Yes	Yes	Yes	Yes
		2000	No(3)	Yes	No <sup>(3)</sup>	Yes	Yes
From	SQL Server	2000	No <sup>(3)</sup>	Yes	No <sup>(3)</sup>	Yes	Yes

**Mpon** specifies the repository format, database engine, and version of OLAP Services or Analysis Services that archives a database; **To** specifies the repository format, database engine, and version of OLAP Services or Analysis Services that restores a database.

2 MDS represents the Meta Data Services (previously named Microsoft Repository) format supported by SQL Server 2000.

3 OLAP servers do not support restoration of Analysis Services databases.

#### See Also

Archiving and Restoring Databases

OLE DB Provider for Jet

## **Microsoft SQL Server 2000 Copyright and Disclaimer**

This document, including sample applications herein, is provided for informational purposes only and Microsoft makes no warranties, either express or implied, in this document. Information in this document, including samples, URL and other Internet Web site references, is subject to change without notice. The entire risk of the use or the results of the use of this document remains with the user.

The primary purpose of a sample is to illustrate a concept, or a reasonable use of a particular statement or clause. Most samples do not include all of the code that would normally be found in a full production system, as a lot of the usual data validation and error handling is removed to focus the sample on a particular concept or statement. Technical support is not available for these samples or for the provided source code.

Unless otherwise noted, the example companies, organizations, products, people, and events depicted herein are fictitious and no association with any real company, organization, product, person, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

©1996-2000 Microsoft Corporation. All rights reserved.

Active Directory, ActiveX, BackOffice, CodeView, Developer Studio, FoxPro, JScript, Microsoft, Microsoft Press, Microsoft SQL Server, MSDN, MS-DOS, Outlook, PivotChart, PivotTable, PowerPoint, Visual Basic, Visual C++, Visual Studio, Win32, Windows 2000, Windows, and Windows NT are either registered

trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Version: 8.00.000

# **Additional SQL Server Resources**

This table provides Internet resources for information about Microsoft® SQL Server<sup>™</sup> and related products and technologies.

Resource	Address
Microsoft Product Support Services Web	http://support.microsoft.com/directory
Microsoft Usenet	news://msnews.microsoft.com/
Microsoft Windows®	http://www.microsoft.com/hcl
Hardware Compatibility List	
MSDN®	http://msdn.microsoft.com
Meta Data Services (formerly known as Microsoft Repository)	http://msdn.microsoft.com
Professional Association for SQL Server	http://www.sqlpass.org/
Microsoft SQL Server Developer Center	http://msdn.microsoft.com
SQL Server Magazine	http://www.sqlmag.com/
Microsoft SQL Server Support	http://support.microsoft.com/support/sql
TechNet Site	www.Microsoft.com/technet
Microsoft Accessibility Web site	http://www.microsoft.com/enable
Microsoft SQL Server Web site	http://www.microsoft.com/sql
Microsoft SQL Server Web site, English Query page	http://www.microsoft.com/sql
Microsoft SQL Server Web site, Analysis Services page	http://www.microsoft.com/sql

 XML Developer Center
 http://www.msdn.microsoft.com/xml/default.asp

#### Backup and Restore (Level 1)

Because backups are not compatible between servers running Microsoft® SQL Server<sup>™</sup> 2000 and servers running earlier versions of SQL Server, SQL Server 6.*x* database dumps (backups) cannot be restored onto a SQL Server 2000 server. For more information about upgrading your databases to SQL Server 2000, see Upgrading Databases from SQL Server 6.5 (Upgrade Wizard).

SQL Server 6.x	SQL Server 2000
The VOLUME clause of the DUMP and LOAD statements indicated the volume ID for a dump device.	The VOLUME keyword has been replaced by the MEDIANAME clause. Use of the VOLUME clause results in an error.
	Remove all references of the VOLUME keyword in all <u>BACKUP</u> , <u>DUMP</u> , <u>LOAD</u> , or <u>RESTORE</u> statements and replace with references to MEDIANAME.
The DUMP and LOAD statements supported the use of diskettes.	Backing up to diskette is not supported. Back up to hard disk, and then copy the backup file to one or more diskettes.
The <b>sysbackuphistory</b> , <b>sysbackupdetail</b> , <b>sysrestorehistory</b> , and <b>sysrestoredetail</b> system tables tracked DUMP and LOAD history information.	The DUMP and LOAD history tracking system tables have been removed and replaced by a new set of system tables. Remove all references to <b>sysbackuphistory</b> , <b>sysbackupdetail</b> , <b>sysrestorehistory</b> , and <b>sysrestoredetail</b> . Because the structure and contents of the backup system tables have changed significantly, familiarize yourself with these new system tables before referencing them:

<u>backupfile, backupmediafamily,</u> <u>backupmediaset, backupset,</u> <u>restorefile, restorefilegroup,</u> and <u>restorehistory</u> .
<u>restoremstory</u> .

#### **Configuration Options (Level 1)**

Administrative scripts may have used these configuration options. For more information about configuration options, see <u>Setting Configuration Options</u>.

SQL Server 6.x	SQL Server 2000
backup buffer size specified the size	Removed; no longer supported.
of the dump and load buffer (used to	Remove all references to <b>backup</b>
increase backup speed).	buffer size.
backup threads specified the	Removed; no longer supported.
number of threads to be reserved for	Remove all references to <b>backup</b>
striped dump and load operations.	threads.
database size set the default number	Removed; no longer supported.
of megabytes (MB) allocated to each	Remove all references to <b>database</b>
new user database.	size.
free buffers determined the	Removed; no longer supported.
threshold of free buffers available to	Remove all references to <b>free</b>
the system.	buffers.
hash buckets set the number of	Removed; no longer supported.
buckets used for hashing pages to	Remove all references to <b>hash</b>
buffers in memory.	buckets.
LE threshold maximum determined	Removed; no longer supported.
the maximum number of page locks	Remove all references to $\mathbf{LE}$
to hold before escalating to a table	threshold maximum.
lock.	
LE threshold minimum determined	Removed; no longer supported.
the minimum number of page locks	Remove all references to $\mathbf{LE}$
required before escalating to a table	threshold minimum.
lock.	
LE threshold percent specified the	Removed; no longer supported.
percentage of page locks needed on a	Remove all references to <b>LE</b>
table before a table lock is requested.	threshold percent.
logwrite sleep specified the number	Removed; no longer supported.
of milliseconds that a write to the log	Remove all references to <b>logwrite</b>

will be delayed if the buffer is not full.	sleep.
<b>max lazywrite IO</b> tuned the priority of batched asynchronous I/O operations performed by the lazy writer.	Removed; no longer supported. Remove all references to <b>max</b> <b>lazywrite IO</b> .
<b>memory</b> set the size of available memory, in 2K units.	Removed; no longer supported. Memory is configured automatically based on need and available memory. To control the range of memory configured automatically, use the <b>min server memory</b> and <b>max server</b> <b>memory</b> options. Remove all references to <b>memory</b> .
<b>open databases</b> set the maximum number of databases that can be open at one time on SQL Server.	Removed; no longer supported. Remove all references to <b>open</b> <b>databases</b> .
<b>procedure cache</b> specified the percentage of memory allocated to the procedure cache after the SQL Server memory needs are met.	Removed; no longer supported. Remove all references to <b>procedure</b> <b>cache</b> .
<b>RA cache hit limit</b> specified the number of cache hits that a read- ahead request could have before it was canceled.	Removed; no longer supported. Remove all references to <b>RA cache</b> <b>hit limit</b> .
<b>RA cache miss limit</b> specified the number of cache misses that occurred during a horizontal traversal before read-ahead started for that command.	Removed; no longer supported. Remove all references to <b>RA cache</b> <b>miss limit</b> .
<b>RA delay</b> specified the delay of read- ahead, in milliseconds.	Removed; no longer supported. Remove all references to <b>RA delay</b> .
<b>RA pre-fetches</b> determined how far ahead the read-ahead manager read (on an extent basis) before the pre- fetch manager idled.	Removed; no longer supported. Remove all references to <b>RA pre-</b> <b>fetches</b> .

<b>RA slots per thread</b> specified the number of simultaneous requests each read-ahead service thread managed.	Removed; no longer supported. Remove all references to <b>RA slots</b> <b>per thread</b> .
<b>RA worker threads</b> specified the number of threads used to service read-ahead requests.	Removed; no longer supported. Remove all references to <b>RA worker</b> <b>threads</b> .
<b>recovery flags</b> determined what information SQL Server displayed in the error log during recovery.	Removed; no longer supported. Remove all references to <b>recovery</b> <b>flags</b> .
<b>remote conn timeout</b> specified a time limit to break a server-to-server connection.	Removed; no longer supported. Remove all references to <b>remote</b> <b>conn timeout</b> .
<b>SMP concurrency</b> specified the maximum number of CPUs that would be used by SQL Server.	Removed; no longer supported. Remove all references to <b>SMP</b> <b>concurrency</b> .
<b>sort pages</b> specified the maximum number of pages to be allocated to sorting per user.	Replaced by <b>min memory per</b> <b>query</b> . For more information about the <b>min memory per query</b> option, see <u>Server Memory Options</u> . Replace all references of <b>sort pages</b> with <b>min memory per query</b> and <b>index create memory</b> .
<b>tempdb in ram</b> placed the <b>tempdb</b> database in RAM, if needed.	No longer supported because SQL Server 2000 has been optimized for maximum performance. Remove all references to <b>tempdb in</b> <b>ram</b> .
<b>trace flag 204</b> supported queries containing sort columns in the ORDER BY clause not included in the <i>select list</i> when the DISTINCT keyword was supplied.	No longer supported. Remove all references to <b>trace flag 204</b> . For more information about supported trace flags, see <u>Trace Flags</u> .
user connections set the maximum	Now an advanced option. Default

	user connections.	
to SQL Server allowed.	growth. Remove all references to	
number of simultaneous connections	value of 0 indicates automatic	

For more information about other changes to configuration options, see SQL Server 2000 and SQL Server version 7.0.

# **Custom Sort Orders (Level 1)**

SQL Server 6.x	SQL Server 2000
Custom sort orders were	Removed; no longer available or
installed from definition files	supported. Remove all references to
(usually with an .srt file	custom sort orders. During installation of
extension).	SQL Server 2000, select an appropriate sort
	order. For more information, see <u>Windows</u>
	Collation Sorting Styles.

# Databases (Level 1)

SQL Server 6.x	SQL Server 2000
The ON <i>database_device</i> = size	If the file was not created originally by
clause of ALTER DATABASE	DISK INIT, the ON <i>database_device</i> =
specified the amount of space, in	<i>size</i> syntax cannot be specified with
megabytes (MB), allocated to	ALTER DATABASE. Instead, use the
the database extension and could	MODIFY FILE clause of ALTER
be used following DISK INIT to	DATABASE to alter the size of a
alter the database device size.	database file. Remove all references of
	the ON <i>database_device</i> = <i>size</i> clause of
	ALTER DATABASE. For more
	information, see <u>ALTER DATABASE</u> .

#### **Database Options (Level 1)**

Administrative scripts may have used these database options. In SQL Server 2000, database options should be set with the ALTER DATABASE statement rather than the **sp\_dboption** stored procedure. For more information about database options, see <u>Setting Database Options</u> and <u>ALTER DATABASE</u>.

SQL Server 6.x	SQL Server 2000
The <b>subscribe</b> option of <b>sp_dboption</b>	Removed; no longer available. Use
enabled or disabled a database for	<pre>sp_addsubscription to enable or</pre>
subscriptions.	disable a database for subscriptions.
The <b>no chkpt. on recovery</b> option of	Removed; no longer available.
<b>sp_dboption</b> defined whether or not a	When using a warm standby server,
checkpoint record was added to a	use the WITH STANDBY clause of
database recovered during a SQL	the <u>RESTORE</u> statement.
Server startup.	

# Data Access Objects (DAO) (Level 1)

SQL Server 6.x	SQL Server 2000
Version 3. <i>x</i> of the Data Access	Because the ODBC driver that ships with
Objects (DAO) functioned	SQL Server 2000 exposes new GUID and
properly when accessing SQL	Unicode data types when connecting to
Server version 6. <i>x</i> servers.	SQL Server, DAO version 3. <i>x</i> does not
	work properly with SQL Server 2000.
	However, the <b><u>odbccmpt</u></b> <u>Utility</u> is
	provided to enable SQL Server version
	6. <i>x</i> ODBC compatibility for a DAO
	application.

# DBCC (Level 1)

SQL Server 6.x	SQL Server 2000
DBCC DBREINDEX used the	Removed; no longer supported. Remove
SORTED_DATA and	all references to either the
SORTED_DATA_REORG	SORTED_DATA or the
clauses. The SORTED_DATA	SORTED_DATA_REORG clauses of
clause eliminated the sort	DBCC DBREINDEX and replace with
performed when a clustered index	references to the DROP_EXISTING
was created and physically	clause of <u>CREATE INDEX</u> .
reorganized the data. The	
SORTED_DATA_REORG clause	
eliminated the sort performed	
when a clustered index was	
created.	
DBCC SHRINKDB either	Removed; no longer supported or
returned the minimum size to	available. Remove all references of
which a database could shrink, or	DBCC SHRINKDB and replace with
shrank the size of the specified	references to <u>DBCC</u>
database to the specified value.	SHRINKDATABASE. Consider
	shrinking databases automatically by
	using the AUTO_SHRINK option of
	ALTER DATABASE.
DBCC MEMUSAGE provided	Removed; no longer supported or
detailed reports on memory use.	available. Remove all references of
	DBCC MEMUSAGE and replace with
	references to these Performance
	Monitor counters.

Performance Monitor object	
name	Performance Monitor counter name
SQL Server: Buffer Manager	Procedure Cache Pages In Use
<u>Object</u>	
	Procedure Cache Size (pages)

<u>SQL Server: Cache Manager</u> <u>Object</u>	Procedure Cache Hit Ratio
	Procedure Cache Pages
	Procedure Cache Object Counts*

\* These counters are available for various categories of cache objects including ad hoc sql, prepared sql, procedures, triggers, and so on.

# **DB-Library (Level 1)**

SQL Server 6.x	SQL Server 2000
DB-Library's two-phase commit	The DB-Library two-phase commit is
special library managed	no longer supported. Use Microsoft
transactions distributed across two	Distributed Transaction Coordinator
or more servers.	(MS DTC) to accomplish simultaneous
	updates on two servers. Remove all
	references to DB-Library's two-phase
	commit.
DB-Library applications could be	The development libraries for DB-
developed in Microsoft® Visual	Library for Visual Basic are not
Basic <sup>®</sup> .	supplied. Existing DB-Library for
	Visual Basic applications will run
	against SQL Server 2000, but must be
	maintained using the development
	libraries for SQL Server 6.5. All new
	Visual Basic applications written to
	access SQL Server should use the Visual
	Basic data APIs such as ActiveX Data
	Objects (ADO) and Remote Data
	Objects (RDO).

# **DECnet Network Library (Level 1)**

SQL Server 6.x	SQL Server 2000
For Intel-based, MIPS-based, and	Removed; no longer supported.
Alpha AXP-based computers, server	Remove all references to the
DECnet Sockets Net-Libraries	DECnet Sockets Net-Libraries.
provided connectivity with	
PATHWORKS networks by allowing	
clients running on VMS to connect to	
SQL Server.	

# Disk Commands (Level 1)

SQL Server 6.x	SQL Server 2000
DISK REINIT and DISK	Removed; no longer supported or
REFIT restored usage	available. Remove all references to DISK
information from system tables	REINIT. Replace all references of DISK
when a device existed (the file	REFIT with references to <u>ALTER</u>
was present) but the entries in	DATABASE, which adds and drops
sysusages no longer existed.	filegroups included in a database, and
	modifies the size of each database
	filegroup.

# Disk Mirroring (Level 1)

SQL Server 6.x	SQL Server 2000
DISK MIRROR, DISK	No longer supported because SQL Server
REMIRROR, and DISK	mirroring is no longer supported. Use
UNMIRROR performed SQL	Microsoft Windows NT® or hardware-
Server disk mirroring.	based RAID. For more information, see
	your Windows NT or hardware
	documentation.

#### Indexes (Level 1)

SQL Server 6.x	SQL Server 2000
The SORTED_DATA_REORG clause of CREATE INDEX eliminated the sort performed when a clustered index was created. The SORTED_DATA clause of CREATE INDEX eliminated the sort performed when a clustered index was created and physically reorganized the data.	Replaced by the DROP_EXISTING clause of CREATE INDEX. Remove all references to the SORTED_DATA_REORG clause of <u>CREATE INDEX</u> and replace with references to DROP_EXISTING. Removed; no longer available. Remove all references to the SORTED_DATA clause of CREATE INDEX.
bcp could import an already sorted data file into a SQL Server table. Creating a clustered index on an ordered table could be optimized by using the SORTED_DATA clause of CREATE INDEX. The SORTED_DATA clause forced SQL Server not to sort or reorganize the previously ordered table.	SQL Server returns an error message stating that the SORTED_DATA clause of CREATE INDEX is ignored and no longer supported. Remove all references to the SORTED_DATA clause of CREATE INDEX. Consider creating the clustered index before using <b>bcp</b> to import the data. <b>bcp</b> uses improved index maintenance strategies to make data importation with a preexisting index faster than earlier releases and avoids resorting of data after importation.
The ALLOW_DUP_ROW and IGNORE_DUP_ROW clauses of the CREATE INDEX statement allowed data to be updated into tables with a unique index and without having to filter out duplicates first.	No longer supported. Using either ALLOW_DUP_ROW or IGNORE_DUP_ROW in the CREATE INDEX statement generates a warning message. If there is no unique clustered index and there is a need to avoid duplicate rows, create a unique

constraint on one or more columns other
than the clustering key.

#### **Open Data Services (Level 1)**

In SQL Server 2000, Open Data Services, now called extended stored procedures, no longer supports gateway applications.

SQL Server 6.x	SQL Server 2000
The ODBC client driver for Open Data Services gateways (ODSGT32.DLL) and associated resource files were used by ODBC clients to connect to Open Data Services gateway servers.	Not shipped with SQL Server 2000. The SQL Server version 6. <i>x</i> ODSGT32.DLL and associated resource files work against an Open Data Services gateway recompiled with SQL Server version 7.0 headers and libraries. Use the SQL Server version 6. <i>x</i> ODBC client driver for Open Data Services (ODSGT32.DLL) and associated resource files to connect from an ODBC client to an Open Data Services gateway. Consider redesigning your application using Windows NT Component Services.
Open Data Services data structures such as SRV_CONFIG, SRV_PROC, and SRV_SERVER were exposed in the Open Data Services header file.	These data structures are no longer exposed, and the data structure members have changed. Applications that reference these data structures directly or their members must be changed and recompiled using the SQL Server 7.0 Open Data Services header file (srv.h) and relinked using the SQL Server 7.0 Open Data Services library file (opends60.lib). These changes should be made to avoid the possibility of server failures.
Earlier versions of SQL Server could make remote stored procedure calls against gateways	SQL Server 2000 does not support remote stored procedure calls against gateways compliant with 6. <i>x</i> and 4. <i>x</i> versions of

compliant with 6. <i>x</i> or 4. <i>x</i>	Open Data Services. SQL Server 2000
versions of Open Data Services.	does support remote stored procedure
	calls against gateways compliant with
	SQL Server 2000. Gateways compiled
	and linked with earlier versions of Open
	Data Services should be recompiled with
	SQL Server 7.0 version of Open Data
	Services. Consider using distributed
	query if your target data source has an
	ODBC or an OLE DB provider on
	Windows NT or Windows 95/98.

#### **Program Group Tools and Utilities (Level 1)**

In Microsoft® SQL Server<sup>™</sup> 2000, these tools have been renamed or replaced.

SQL Server 6.x	SQL Server 2000
ISQL_w	SQL Query Analyzer
MS Query	N/A
SQL Client Configuration	Client Network Utility
SQL Enterprise Manager	SQL Server Enterprise Manager
SQL Help	N/A
SQL Security Manager	N/A
SQL Trace	SQL Server Profiler
SQL Performance	N/A. SQL Server performance counters are
Monitor	added to the Windows 2000 System Monitor or
	the Windows NT 4.0 Performance Monitor.
SQL Service Manager	SQL Server Service Manager
SQL Setup	N/A

# **Replication (Level 1)**

SQL Server 6.x	SQL Server 2000
Restricted publications could be	Restricted publications cannot be
created through the user interface	created through the user interface and
and used in replicating data.	are no longer supported. Remove all
	references to restricted publications. A
	replacement for restricted publications
	will be available in a later release. For
	more information, see <u>Replication</u>
	<u>Overview</u> .
Publish and subscribe properties	No longer available. Remove all
could be set using the <b>DBOption</b>	references to the <b>DBOption</b> object and
object.	replace with references to the
	EnablePublishing property of the
	ReplicationDatabase object.
The <b>repl_publisher</b> login allowed	No longer available. Remove all
replication processes on the	references to the <b>repl_publisher</b> login
distributor to connect to a	and replace with references to a login in
subscription server and replicated	a publication access list (PAL). For
table schema and data to	more information about PALs, see
destination databases.	Publication Access Lists.

# Security (Level 1)

SQL Server 6.x	SQL Server 2000
DENY was not a reserved	DENY is a reserved keyword. Rename
keyword and could be used as	any object named DENY. Change all
an object identifier.	Transact-SQL statements and scripts
	referencing the object to use the new
	object name. If DENY is retained as an
	object identifier, all references to the
	object must use <u>Delimited Identifiers</u> .

#### Segments (Level 1)

SQL Server 6.x	SQL Server 2000
Indexes could be placed on segments using the CREATE	Segments are no longer supported. However, CREATE INDEX can create
INDEX statement.	an index on a filegroup. Remove all references to segments and replace with references to filegroups within a <u>CREATE INDEX</u> statement.
Tables could be created on a particular segment by using the CREATE TABLE statement.	CREATE TABLE references files and filegroups instead of segments. Remove all references to segments and replace with references to files and filegroups within a <u>CREATE TABLE</u> statement.
User-defined segments allowed the placement of database objects on certain devices for performance reasons.	Segments are no longer supported. Multidisk RAID devices generally provide a greater increase in performance with a lower associated administrative cost. Use filegroups for user-defined placement of data, indexes, or text. Remove all references to these segment-related system stored procedures: <b>sp_addsegment</b> <b>sp_extendsegment</b> <b>sp_extendsegment</b> <b>sp_helpsegment</b> Create, modify, or drop files and filegroups; and place indexes on files or filegroups using <u>CREATE TABLE</u> , <u>CREATE DATABASE</u> , ALTER <u>DATABASE</u> , and <u>CREATE INDEX</u> .

# Services (Level 1)

SQL Server 6.x	SQL Server 2000
SQL Executive provided the SQL	SQL Executive tasks are now performed
Server scheduling engine. SQL	by SQL Server Agent. Use SQL Server
Executive offered extensive and	Agent for scheduling purposes. For
varied task scheduling and	more information, see <u>Configuring the</u>
alerting abilities, and was capable	<u>SQL Server Agent Service</u> .
of handling large client/server	
environments.	

#### SET DISABLE\_DEF\_CNST\_CHK (Level 1)

SQL Server 6.x	SQL Server 2000
The SET	Removed; no longer available. Remove
DISABLE_DEF_CNST_CHK	all references to SET
setting controlled interim	DISABLE_DEF_CNST_CHK.
constraint checking.	

#### SET SHOWPLAN (Level 1)

SQL Server 6.x	SQL Server 2000
SET SHOWPLAN generated a	SET SHOWPLAN has been replaced with
description of the processing	SET SHOWPLAN_ALL and SET
plan for the query and	SHOWPLAN_TEXT. The SET
processed it immediately	SHOWPLAN_ALL and SET
unless the SET NOEXEC	SHOWPLAN_TEXT statements return
setting was enabled.	only query or statement execution plan
	information and do not execute the query
	or statement. To execute the query or
	statement, turn the appropriate showplan
	statement OFF. The query or statement will
	then execute.
	Remove all references to either SET
	SHOWPLAN ON or SET SHOWPLAN
	OFF and replace with references to either
	SET SHOWPLAN_ALL ON, SET
	SHOWPLAN_TEXT ON, SET
	SHOWPLAN_ALL OFF, or SET
	SHOWPLAN_TEXT OFF. Expect
	differences in behavior as compared to
	earlier versions of SQL Server.

# SQL Alerter (Level 1)

SQL Server 6.x	SQL Server 2000
SQL Alerter, SQLALRTR.exe, was used to integrate the alert engine with the Windows NT Performance Monitor alerter.	Removed; no longer supported or available. Replaced by SQL Server performance condition alerts. Remove all references to SQL Alerter and replace with references to SQL Server performance condition alerts. For more information, see <u>Defining Alerts</u> .

#### SQL-DMO (Level 1)

SQL-DMO applications are administrative tools and should be updated to work with SQL Server 2000. It is recommended that code be recompiled and any error messages returned from the build process be used to track any necessary changes.

SQL Server 6.x	SQL Server 2000
SQL-DMO is implemented in Sqlole.dll. The SQL-DMO objects exhibit properties, methods, and events that automate administrative tasks for SQL Server version 6.5 and earlier. SQL-DMO, implemented in Sqlole.dll, cannot connect to and operate against SQL Server 2000 or SQL Server 7.0.	SQL-DMO is implemented in Sqldmo.dll. These SQL-DMO objects expose the properties, methods, and events that automate administrative tasks for SQL Server. They cannot be used to connect to and operate against a SQL Server version 6.5 (or earlier) server. Therefore, it is recommended that you rewrite SQL-DMO applications.
	If the application must operate against both SQL Server 2000 and version 6.5 or earlier of SQL Server, reference both Sqldmo.dll and Sqlole.dll components in the application. Develop new, separate subroutines referencing the SQL Server 2000 SQL-DMO objects from the existing subroutines.
	If the application will work against SQL Server 2000 only, rewrite existing subroutines to reference SQL Server 2000 SQL-DMO objects.
	If the application will not be used against your new SQL Server 2000 server(s), continue to use the application unchanged.

# System Stored Procedures (General Extended Procedures) (Level 1)

SQL Server 6.x	SQL Server 2000
<b>xp_snmp_getstate</b> returned the state	Removed; no longer available.
of the SQL Server Simple Network	Remove all references to either
Management Protocol (SNMP)	<b>xp_snmp_getstate</b> or
agent. <b>xp_snmp_raisetrap</b> permitted	xp_snmp_raisetrap.
a client to define and send a trap (an	
SNMP alert) to an SNMP client.	

#### System Stored Procedures (Replication) (Level 1)

SQL Server 6.x	SQL Server 2000
<b>sp_replica</b> remotely set (on a Subscriber) a <b>sysobjects</b> category bit that marked the table as a replica.	Removed; no longer supported or available. Remove all references to <b>sp. replica</b> .
<b>sp_replsync</b> acknowledged completion of a manual synchronization when used from a Subscriber.	Removed; no longer supported or available. Remove all references to <b>sp_replsync</b> .
<b>sp_helppublicationsync</b> provided information about a scheduled synchronization task for a publication.	No longer supported. An error message is returned if this stored procedure is used. Remove all references to <b>sp_helppublicationsync</b> .
<b>sp_subscribe</b> and <b>sp_unsubscribe</b> remotely added or canceled a subscription to a particular article within a publication, to a whole publication, or to all publications.	No longer supported. An error message is returned if this stored procedure is used. Remove all references to either <b>sp_subscribe</b> or <b>sp_unsubscribe</b> , or use the <u>@@ERROR</u> function to test for errors.
<b>name</b> <i>value</i> parameter of <b>sp_changepublication</b> was used to provide the new publication name.	Removed; no longer supported or available. Remove all references to the <b>name</b> <i>value</i> parameter of <u>sp_changepublication</u> .
<pre>sp_addpublisher added a Publisher at the Subscriber and added a Distribution Publisher at the Distributor. sp_droppublisher dropped a publication server.</pre>	Replaced by <b>sp_adddistpublisher</b> . Remove all references to <b>sp_addpublisher</b> and replace with references to <b>sp_adddistpublisher</b> . Removed; no longer supported or available. Remove all references to <b>sp_droppublisher</b> . To drop a Publisher at a Distributor, use

	sp_dropdistpublisher.
<b>sp_distcounters</b> was used to query	No longer supported. The new view
for delivered or undelivered	MSdistribution_status presents
commands as used by Performance	much of the same information.
Monitor, which no longer uses this	
procedure.	
<b>sp_helpreplicationdb</b> was used to	Removed; no longer supported.
return information about a specified	Remove all references to
database or a list of all publication	<b>sp_helpreplicationdb</b> and replace
databases on the server.	with references to
	sp_helpreplicationdboption.
<b>sp_replstatus</b> updated the internal	Removed; no longer supported.
table structure for replication.	Remove references to <b>sp_replstatus</b> .

SQL Server 6.x	SQL Server 2000
Several system stored procedures were used for documenting keys.	Removed; no longer supported or available. Use declarative referential integrity by implementing keys and constraints with either ALTER TABLE or CREATE TABLE. Remove all references to these system stored procedures and replace with references to either sp_help or sp_helpconstraint: sp_commonkey sp_dropkey sp_foreignkey sp_helpjoins sp_helpkey sp_primarykey
<b>sp_placeobject</b> put future space allocations for a table or index on a particular segment.	<b>sp_placeobject</b> is no longer available because segments no longer exist. Use the ON FILEGROUP syntax of the CREATE TABLE statement to place table or index information about a separate filegroup. Remove all references of <b>sp_placeobject</b> and replace with references to the ON FILEGROUP clause of the <u>CREATE TABLE</u> statement.
<b>sp_dbinstall</b> installed a database and its devices, and was used for removable media.	Removed; no longer supported or available. Remove all references to <b>sp_dbinstall</b> and replace with references to <u>sp_attach_db</u> .

# System Stored Procedures (System) (Level 1)

<pre>sp_makestartup and sp_unmakestartup set a stored procedure for auto execution and discontinued auto execution of the stored procedure, respectively. The sp_helplogins, sp_helprotect, and sp_tableoption system stored procedures supported pattern matching (using wildcard characters), which allowed flexibility in specific parameters.</pre>	Removed; no longer supported or available. Remove all references of either <b>sp_makestartup</b> or <b>sp_unmakestartup</b> and replace with references to <b>sp_procoption</b> . Pattern matching using the wildcard characters is no longer supported in these system stored procedures because any system stored procedure identifier may contain a pattern matching character. Remove all references to pattern matching in <b>sp_helplogins, sp_helprotect</b> , and
The <b>fallback</b> option of <b>sp_serveroption</b> indicated a fallback server.	<pre>sp_tableoption. The fallback option of sp_serveroption is no longer available because the fallback option is no longer supported. Remove all references to the fallback option of sp_serveroption.</pre>
<b>sp_setlangalias</b> assigned or changed the alias for an alternate language.	Removed; no longer supported. Use the aliases provided in <b>syslanguages</b> . Remove all references to <b>sp_setlangalias</b> .
<b>sp_droplanguage</b> dropped an alternate language from the server and removed its row from <b>master.dbo.syslogins</b> .	Removed; no longer supported. Remove all references to <b>sp_droplanguage</b> .
Fallback support was provided by executing system stored procedures that shifted control of databases and devices from a broken primary server to a fallback server.	Fallback support is no longer supported using the fallback system stored procedures. Support for fallback servers is supported using Microsoft Windows NT Clustering Service. Remove all references to these fallback system stored procedures:

	sp_fallback_activate_svr_db, sp_fallback_deactivate_svr_db, sp_fallback_enroll_svr_db, sp_fallback_help, sp_fallback_permanent_svr, sp_fallback_upd_dev_drive, sp_fallback_withdraw_svr_db.
<b>sp_devoption</b> displayed or set device status.	Removed; no longer available. Remove all references to <b>sp_devoption</b> .
<b>sp_diskdefault</b> set a database device status to indicate whether the device can be used for database storage when the user does not specify a database device or specifies DEFAULT with the CREATE DATABASE or ALTER DATABASE statements.	Removed; no longer available. Remove all references to <b>sp_diskdefault</b> .
<b>sp_helplog</b> reported the name of the device that contains the first page of the log in the current database.	Removed; no longer available. Remove all references to <b>sp_helplog</b> .
<b>sp_helpstartup</b> reported a listing of all auto-start stored procedures.	Removed; no longer available. Remove all references to <b>sp_helpstartup</b> and replace with references to <b>sp_procoption</b> .
<b>sp_sqlexec</b> provided a convenient way for SQL Server database clients and servers to send a language statement of any format to an Open Data Services server application.	Removed; no longer available. Remove all references to <b>sp_sqlexec</b> .
<b>sp_helprevdatabase</b> analyzed an existing database and created a script that could be used to replicate the database structure on another server.	Removed; no longer available. If applicable, use the SQL-DMO <u>Script</u> <u>Method</u> of the <u>Database Object</u> to generate similar information. Remove all references to <b>sp_helprevdatabase</b> .

<b>sp_addlanguage</b> added an alternate	Removed; no longer available.
language to a server.	Remove all references to
	sp_addlanguage.

#### System Stored Procedures (Tasks) (Level 1)

Replace the following unsupported Microsoft® SQL Server<sup>™</sup> 6.*x* task-related system stored procedures with the corresponding SQL Server 2000 job-related system stored procedures.

SQL Server 6.x	SQL Server 2000
sp_addalert	sp_add_alert
sp_addnotification	sp_add_notification
sp_addoperator	sp_add_operator
sp_dropalert	sp_delete_alert
sp_dropnotification	sp_delete_notification
sp_dropoperator	sp_delete_operator
sp_helpalert	sp_help_alert
sp_helphistory	sp_help_jobhistory
sp_helpnotification	<pre>sp_help_notification</pre>
sp_helpoperator	sp_help_operator
sp_purgehistory	<u>sp_purge_jobhistory</u>
sp_runtask	<u>sp_start_job</u>
sp_stoptask	<u>sp_stop_job</u>
sp_updatealert	sp_update_alert
sp_updatenotification	<pre>sp_update_notification</pre>
sp_updateoperator	sp_update_operator

Task management has been changed to job management.

# System Tables (Level 1)

SQL Server 6.x	SQL Server 2000
SQL Server 6.x System tables were used internally by SQL Server for a wide range of uses, including maintaining the list of character sets that SQL Server could use and containing information about active locks.	System tables have changed significantly. Most SQL Server 6. <i>x</i> system tables will continue to work properly. Views provided allow applications referencing SQL Server 6. <i>x</i> system tables to continue functioning properly. However, some SQL Server 2000 data cannot be referenced through these views. Use the provided Information Schema Views or ODBC catalog system stored procedures to obtain system table information. Modify scripts as appropriate. Any scripts referencing SQL Server 6. <i>x</i> system tables will not be converted
<b>sysdevices</b> contained one row for each disk dump, tape dump, and database device.	properly. The <b>mirrorname</b> and <b>stripeset</b> columns have been removed. <b>sysdevices</b> is retained only for dump devices and also for backward compatibility (supporting DISK INIT and SQL Server 6. <i>x</i> CREATE DATABASE syntax). Remove all references to the <b>mirrorname</b> and <b>stripeset</b> columns of <u>sysdevices</u> .
<b>syshistory</b> contained one row for each scheduled event, alert, or task that occurred.	Replaced by <b>sysjobhistory</b> . Remove all references to <b>syshistory</b> and replace with references to <b>sysjobhistory</b> .
<b>sysindexes</b> contained one row for each clustered index and one row for each nonclustered index.	The <b>distribution</b> , <b>segment</b> , <b>rowpage</b> , <b>keys1</b> , and <b>keys2</b> columns have been removed. Remove all references to the <b>distribution</b> , <b>segment</b> , <b>rowpage</b> , <b>keys1</b> , and <b>keys2</b> columns of <b>sysindexes</b> . In addition, <b>soid</b> is <b>reserved3</b> , and <b>csid</b> is

	reserved4.
syskeys used for objects	Removed; no replacement. Remove all references to <b>syskeys</b> .
syslocks contained information	Removed; replaced by <b>syslockinfo</b> .
about active locks.	Remove all references to <b>syslocks</b> and
	replace with references to <b>syslockinfo</b> .
syslogs contained the	Removed; no replacement. The database
transaction log.	log is now an operating system file.
	Remove all references to <b>syslogs</b> .
sysprocesses contained	The <b>gid</b> column has been removed. In SQL
information about SQL Server	Server, the <b>suid</b> column has been removed.
processes.	Remove all references to these columns of
	sysprocesses.
sysprocedures contained	Removed; replaced by <b>syscomments</b> . SQL
entries for each view, default,	Server obtains procedure text from
rule, trigger, CHECK	syscomments when procedures need to be
constraint, DEFAULT	compiled. Remove all references to
constraint, and stored	sysprocedures and replace with references
procedure.	to <u>syscomments</u> .
syssegments contained one	Removed; no replacement. Segments are
row for each segment (named	no longer supported. Use filegroups
collection of disk fragments).	instead. Remove all references to
	<b>syssegments</b> . Use filegroups instead by
	using <u>CREATE DATABASE</u> , <u>ALTER</u>
	DATABASE, <u>CREATE TABLE</u> , <u>ALTER</u>
	TABLE, and <u>CREATE INDEX</u> .
5	Removed; replaced by <b>sysjobs</b> ,
every scheduled task.	sysjobsteps, and sysjobservers. Remove
	all references to <b>systasks</b> and replace with
	references to <u>sysjobs</u> , <u>sysjobsteps</u> , and
	sysjobservers as appropriate.
sysusages contained one row	Removed; no replacement. SQL Server
for each disk-allocation piece	relies on <b>sysdevices</b> for database file
assigned to a database.	information. Filegroups are supported, and
	the <b>sysfiles</b> and <b>sysfilegroups</b> system
	tables are added. These system tables

	reside in every database and describe database files and filegroups. Remove all references to <b>sysusages</b> .
master.dbo.spt	Removed; no replacement. Remove all
_datatype_info	references to
	master.dbo.spt_datatype_info.

For more information, see <u>System Tables</u>.

# Transactions (Level 1)

# **Utilities (Level 1)**

SQL Server 6.x	SQL Server 2000
The <b>probe</b> login, which required no	The <b>probe</b> login has been eliminated.
password, was used by DB-Library	Windows NT Performance Monitor
and Windows NT Performance	will always use Windows NT
Monitor. The DB-Library two-phase	Authentication, known earlier as
commit library used the <b>probe</b> login	integrated security, to connect to
to check on the status of distributed	SQL Server. Ensure that your
transactions. It was also used by	Windows NT username and
Windows NT Performance Monitor	password have the appropriate
to get statistics from SQL Server.	privileges to use Windows NT
	Performance Monitor.

#### Backup and Restore (Level 2)

SQL Server 6.x	SQL Server 2000
Using the SKIP and INIT clauses of the DUMP statement together overwrote the contents of the backup device unconditionally.	The SKIP and INIT clauses of the BACKUP statement preserve the Microsoft Tape Format media header. In some situations, this prevents overwriting the backup contents. The FORMAT clause overwrites the media unconditionally, generating a new header, and is required for media used for the first time or when necessary to overwrite the media header. Expect different results as compared to earlier versions of SQL Server. If the media is empty, SKIP and INIT act the same as the FORMAT clause of the
	<b>BACKUP</b> or DUMP statements and write a new media header. If the media is not empty, SKIP and INIT do not write a new media header.
The LOAD statement did not create the database automatically when restoring the database backup.	It is no longer necessary to create the database before restoring it. The RESTORE statement re-creates the database automatically, including all files. However, database devices are not re-created in <b>sysdevices</b> . These devices are supported only for backward compatibility. After restoration, databases originally created using devices (DISK INIT) appear as if they had been created using SQL Server 2000 file syntax.

	Expect different results as compared to earlier versions of SQL Server. Consider using the new syntax in <u>CREATE</u> <u>DATABASE</u> and <u>ALTER DATABASE</u> for specifying files.
The NO_LOG clause of DUMP was used only when you ran out of space in the database and could not use DUMP TRANSACTION WITH TRUNCATE_ONLY to purge the log. The NO_LOG clause removes the inactive part of the log without making a backup copy of it, and saves space by not logging the operation. The TRUNCATE_ONLY clause of the DUMP statement removed the inactive part of the log without making a backup copy of it.	The NO_LOG and TRUNCATE_ONLY clauses of RESTORE are synonyms. Both clauses of BACKUP now remove the inactive part of the log without making a backup copy of it and truncate the log. Expect different results as compared to earlier versions of SQL Server. Expect the NO_LOG and TRUNCATE_ONLY clauses of the BACKUP or DUMP statements to behave identically.
Recovery of multiple transaction logs could be performed without special keywords in the LOAD statement.	It is no longer possible to restore multiple transaction logs without using the WITH clauses of the RESTORE statement. Expect different results as compared to earlier versions of SQL Server. Use the appropriate <u>RESTORE</u> syntax for restoring a database with multiple transaction logs as shown in the following examples. All but the last RESTORE statement should specify the NORECOVERY clause.
When loading a database, all database options of <b>sp_dboption</b> were unaffected and had to be set manually.	Changes to all <b>sp_dboption</b> database settings (except the <b>offline</b> , <b>merge</b> <b>publish</b> , <b>published</b> , and <b>subscribed</b> settings) are logged, like any other

change. When a database is restored and recovered, all database options of <b>sp_dboption</b> are rolled forward. Every database option will be in its expected state at the time when recovery finished, consistent with the remainder of the
database. Expect different results as compared to earlier versions of SQL Server. It is no longer necessary to reset the database options after a RESTORE operation.

#### Examples

# A. Restore a database by applying a full database backup and multiple transaction logs

This example restores a database with multiple transaction log backups.

RESTORE DATABASE mydb FROM mydb WITH NORECOVERY

RESTORE LOG mydb FROM mydb\_log1 WITH NORECOVERY

RESTORE LOG mydb FROM mydb\_log2 WITH RECOVERY

SQL Server 6.x	SQL Server 2000
A warm standby server could be	A warm standby server can be brought
brought up in read-only mode	up in read-only mode between
between recovery of each	transaction log restore operations if an

transaction log, provided that the <b>no chkpt. on recovery</b> option of <b>sp_dboption</b> was enabled.	undo file is used. Expect different results as compared to earlier versions of SQL Server. Use an undo file for a warm standby server using the STANDBY clause of <u>RESTORE</u> , as shown in the following example.
	example.

#### B. Restore a database using the STANDBY clause and an undo file

This example brings the server up to allow write operations on the databases by using a final, necessary RESTORE statement.

RESTORE DATABASE mydatabase FROM mydb\_backup WITH NORECOVERY

RESTORE LOG mydb FROM mydb\_log1 WITH RECOVERY STANDBY (FILENAME = 'c:\mssql\data\mydbu

RESTORE LOG mydb FROM mydb\_log2 WITH RECOVERY STANDBY (FILENAME = 'c:\mssql\data\mydbu

RESTORE DATABASE mydb WITH RECOVERY

### Bulk Copy (Level 2)

SQL Server 6.x	SQL Server 2000
The <b>bcp</b> utility (using DB- Library) could import <b>datetime</b> or <b>smalldatetime</b> values in character-mode data files using:	The <b>bcp</b> utility (which uses ODBC) can import <b>datetime</b> and <b>smalldatetime</b> values in character-mode data files using:
• The default format used by DB-Library (mmm dd yyy hh:mmXX where XX is either A.M. or P.M.).	<ul> <li>The default format used by DB-Library.</li> <li>The format used by ODBC (yyyy-mm-dd hh:mm:ss[.f]).</li> </ul>
<ul> <li>Any format supported by dbconvert except the ODBC format.</li> <li>bcp exported character-mode data files with datetime and smalldatetime values by using the default DB-Library format.</li> </ul>	<ul> <li>However, bcp does not use other formats supported by dbconvert.</li> <li>bcp exports datetime and smalldatetime values using the ODBC default format.</li> <li>Expect different results as compared to earlier versions of SQL Server. To bulk copy data in character mode between SQL Server 6.<i>x</i> and SQL Server 2000 servers, use the same bcp version (either SQL Server 6.<i>x</i> or SQL Server 2000) for both importing and exporting data.</li> <li>To export data from a SQL Server 2000 server into a character-mode data file and later import that data using a DB-Library bulk copy application, use the SQL Server 6.<i>x</i> version of bcp.</li> <li>For existing datetime or smalldatetime values in a character-mode data file in a</li> </ul>

	format other than the DB-Library default:
	<ul> <li>Change the values to the DB- Library default format for continued use with SQL Server 6.<i>x</i> and SQL Server 2000 bcp.</li> </ul>
	• Change the values to the ODBC format for use with SQL Server 2000 <b>bcp</b> .
<b>bcp</b> exported <b>money</b> values in	<b>bcp</b> exports <b>money</b> values in character
character mode data files using	mode data files without digit grouping
digit grouping symbols (for	symbols, but with four digits after the
example, the comma in the United	decimal point.
States when using the U.S. version of SQL Server, the US version of Microsoft Windows	Expect different results as compared to earlier versions of SQL Server.
NT, and US settings) and two	To read character files created by
digits after the decimal point.	version 6. <i>x</i> DB-Library <b>bcp</b> in SQL
	Server 2000, use the <b>-V</b> switch. For
	more information, see <u><b>bcp</b> Utility</u> .

#### **Configuration Options (Level 2)**

Administrative scripts may have used these configuration options. For more information about configuration options, see <u>sp\_configure</u> and <u>Setting</u> <u>Configuration Options</u>.

SQL Server 6.x	SQL Server 2000
<b>open objects</b> set the maximum number of database objects that can be open at one time on SQL Server.	Now an advanced option. Default value of 0 indicates automatic growth. Expect different results as compared to earlier versions of SQL Server. Remove all references to <b>open objects</b> . For more information, see <b>open objects</b> Option.
<b>user connections</b> set the maximum allowed number of simultaneous connections to SQL Server.	Now an advanced option. Default value of 0 indicates automatic growth. Expect different results as compared to earlier versions of SQL Server. Remove all references to <u>user connections</u> <u>Option</u> .

### Database Pages and Extents (Level 2)

SQL Server 6.x	SQL Server 2000
A database extent consisted of eight 2 KB pages.	A database extent consists of eight 8 KB pages. Different objects can now share an extent or an object can have its own extent. A table and index both have a minimum of two pages. Expect different results as compared to
	earlier versions of SQL Server. Adjust disk space requirements for adequate database storage. For more information, see <u>Pages and</u> <u>Extents</u> .

### Data Types (Level 2)

SQL Server 6.x	SQL Server 2000
Conversion of <b>binary</b> or <b>varbinary</b> to <b>decimal</b> or <b>numeric</b> was explicit.	This conversion is implicit. Expect different results as compared to earlier versions of SQL Server. Expect conversions of <b>binary</b> or <b>varbinary</b> to <b>decimal</b> or <b>numeric</b> to be implicit. For more information about data type conversions, see <u>CAST and CONVERT</u> . For more information about system- supplied data types, see <u>Data Types</u> .
Conversion of <b>binary</b> or <b>varbinary</b> to <b>smallmoney</b> was not allowed.	This conversion is allowed. Expect different results as compared to earlier versions of SQL Server. Use as appropriate.
Conversion of <b>datetime</b> or <b>smalldatetime</b> to <b>decimal</b> , <b>numeric, float, real, int,</b> <b>smallint, tinyint, money</b> , <b>smallmoney</b> , or <b>bit</b> was not allowed.	This conversion is allowed. Expect different results as compared to earlier versions of SQL Server. Use as appropriate.
Conversion of <b>float</b> or <b>real</b> to <b>binary</b> or <b>varbinary</b> was not allowed.	This conversion is allowed. Expect different results as compared to earlier versions of SQL Server. Use as appropriate.
Conversion of <b>money</b> or <b>smallmoney</b> to <b>char</b> or <b>varchar</b> was implicit.	This conversion is explicit. Expect different results as compared to earlier versions of SQL Server. Expect conversions of <b>money</b> or <b>smallmoney</b> to <b>char</b> or <b>varchar</b> to be explicit.

Conversion of <b>bit</b> to <b>money</b> or <b>smallmoney</b> was not allowed.	This conversion is allowed. Expect different results as compared to earlier versions of SQL Server. Use as appropriate.
<b>sysname</b> data type was <b>varchar(30)</b> .	<b>sysname</b> data type is <b>nvarchar(128)</b> , which allows for 128 Unicode characters. Expect different results as compared to earlier versions of SQL Server. Expect any columns or local variables defined as <b>sysname</b> to allow Unicode data. For more information about <b>sysname</b> , see <u>Data</u> <u>Types</u> .
No direct support for the <b>nchar</b> , <b>nvarchar</b> , and <b>ntext</b> Unicode data types because maximum storage was 255 bytes for <b>char</b> , <b>binary</b> , <b>varchar</b> , and <b>varbinary</b> data types.	The maximum number of bytes that can be stored in <b>char</b> , <b>binary</b> , <b>varchar</b> , and <b>varbinary</b> data types is increased to 8,000. SQL Server 2000 clients fully support the <b>nchar</b> , <b>nvarchar</b> , and <b>ntext</b> data types. SQL Server 6. <i>x</i> clients accessing SQL Server 2000 with these Unicode data types will experience these results: <b>nvarchar</b> data is returned as <b>varchar</b> and <b>nchar</b> data is returned as <b>varchar</b> and <b>nchar</b> data is returned as <b>char</b> . <b>nvarchar</b> and <b>nchar</b> values longer than 255 double- byte characters are truncated to 255 single-byte characters. Attempting to access <b>ntext</b> data causes SQL Server to issue a 4004 error. <b>ntext</b> data cannot be sent to version 6. <i>x</i> clients. <b>char</b> , <b>varchar</b> , <b>binary</b> , and <b>varbinary</b> values longer than 255 bytes are truncated to 255 bytes.

	Expect different results as compared to earlier versions of SQL Server. Expect changes in data when accessing SQL Server 2000 data from version 6. <i>x</i> clients. To eliminate these differences, upgrade the clients to SQL Server 2000 client software.
Using CONVERT to convert an empty string to <b>int</b> (CONVERT(int, ")) or <b>float</b> (CONVERT(float, ")) returned a zero.	Using CAST or CONVERT to convert an empty string to <b>int</b> (CAST(" AS int)) or <b>float</b> (CAST(" AS float)) returns an error message. Expect different results as compared to earlier versions of SQL Server.

### **DB-Library (Level 2)**

SQL Server 6.x	SQL Server 2000
When connected to a version 6. <i>x</i>	A call to <b>dbcursorfetchex</b> resulting in a
SQL Server, a call to	cursor position after the end of the
dbcursorfetchex resulting in a	cursor result set returns SUCCEED. All
cursor position after the end of the	row status indicators are set to 0. This
cursor result set returned:	behavior applies to all types of cursors.
• FAIL, with either a keyset or an insensitive cursor.	Expect different results as compared to earlier versions of SQL Server. Expect different results, compared to SQL Server 6. <i>x</i> , when a call to
• SUCCEED, with all row status indicators set to 0, with either a dynamic or forward cursor.	<b>dbcursorfetchex</b> results in a cursor position after the end of the cursor result set. To achieve SQL Server 6. <i>x</i> behavior, use compatibility level 65.

### Empty Strings (Level 2)

SQL Server 6.x	SQL Server 2000
An empty string could be	Interpretation of an empty string is
interpreted as either a NULL or a	controlled by the compatibility level,
single blank character.	which is set with the <b>sp_dbcmptlevel</b>
	system stored procedure. If the
	compatibility level is 65 or lower, SQL
	Server interprets empty strings as single
	spaces. If the compatibility level is 70 or
	80, SQL Server interprets empty strings
	as empty strings. For more information,
	see <u>sp_dbcmptlevel</u> .
	Expect differences in interpretation of
	empty strings compared to earlier
	versions of SQL Server. Transact-SQL
	functions and statements affected by the
	setting of <u>sp_dbcmptlevel</u> include
	CHARINDEX, DATALENGTH, LEFT,
	LTRIM, PATINDEX, REPLICATE,
	RIGHT, RTRIM, SPACE, SUBSTRING,
	and <u>UPDATETEXT</u> .

### Indexes (Level 2)

SQL Server 6.x	SQL Server 2000
The DROP INDEX statement dropped the pages holding the clustered index B-tree when used on a clustered index.	On a clustered index, the DROP INDEX statement must rebuild all nonclustered indexes. SQL Server must also replace the clustered index keys in the nonclustered leaf rows with row pointers.
	Expect different results as compared to earlier versions of SQL Server. Use the DROP_EXISTING clause of the CREATE INDEX statement if, for example, dropping or re-creating clustered indexes to set a new fill factor value.

### **INSERT (Level 2)**

SQL Server 6.x	SQL Server 2000
An INSERT <b>x</b> SELECT INTO <b>Y</b> statement ignored table <b>Y</b> and inserted the SELECT results into table <b>X</b> , as shown. INSERT X SELECT select_list INTO Y	The INSERTSELECT INTO syntax is retained only when the compatibility setting is equal to 60 or 65. If the compatibility setting is 70 or 80 and a similar query is executed, SQL Server returns a syntax error.
	Expect different results as compared to earlier versions of SQL Server. Set the compatibility setting to 60 or 65 by executing <b>sp_dbcmptlevel</b> to retain version 6. <i>x</i> functionality for queries using INSERTSELECT INTO syntax. Otherwise, use a compatibility setting of 80.
In an INSERT statement, a SELECT statement returning a scalar value was allowed in the VALUES clause.	The INSERT statement cannot have a SELECT statement in the VALUES clause as one of the values to be inserted. The version 6. <i>x</i> supportability is available only when the compatibility setting is equal to 60 or 65.
	Expect different results as compared to earlier versions of SQL Server. Set the compatibility setting to 60 or 65 by executing <b>sp_dbcmptlevel</b> to retain version 6. <i>x</i> functionality for using a SELECT statement in the VALUES clause of an <b>INSERT</b> statement. Otherwise, use a compatibility setting of 80.

A ROLLBACK statement in a stored procedure referenced in an INSERT table EXEC procedure statement caused the INSERT to be rolled back, but the batch continued.	A ROLLBACK statement in the stored procedure referenced by an INSERTEXEC statement causes the entire transaction to be rolled back and the batch stops executing. The version 6. <i>x</i> supportability is available only when the compatibility setting is equal to 60 or 65. Expect different results as compared to earlier versions of SQL Server. Set the compatibility setting to 60 or 65 by executing <b>sp_dbcmptlevel</b> to retain version 6. <i>x</i> functionality for ROLLBACK statement behavior inside an <b>INSERT</b> EXEC statement. Otherwise, use a compatibility setting of 80.
--	---

### Keyset Cursors (Level 2)

SQL Server 6.x	SQL Server 2000
When using a keyset cursor, a row deletion followed by a row insertion using the same key as the deleted row caused the	When using a keyset cursor, a row deletion followed by a row insertion with the same key as the deleted row allows the original row to remain empty and the newly inserted row to be inserted at the end. Expect different results as compared to earlier versions of SQL Server. Expect a change in behavior when inserting and deleting rows with the same key values when using keyset cursors.

### LTRIM and RTRIM Trimming Functions (Level 2)

SQL Server 6.x	SQL Server 2000
The LTRIM and RTRIM functions returned	Zero-length strings are
NULL in queries using zero-length strings:	supported. The queries shown
SELECT RTRIM(")	return nonnull values; the first
	returns " and the second
SELECT DATALENGTH(RTRIM("))	returns 0.
	Expect different results as
	compared to earlier versions of
	SQL Server. <u>LTRIM</u> and
	RTRIM provide different
	output from earlier versions of
	SQL Server.

### ODBC (Level 2)

SQL Server 6.x	SQL Server 2000
SQL_ERROR was returned by	SQL_SUCCESS_WITH_INFO is
SQLExecute, SQLExecDirect, or	returned when an ODBC 3. <i>x</i>
SQLParamData when extended	application uses the ODBC SQL
stored procedures or batches met the	Server 3.51-compliant driver
following criteria:	included with this release (using
• The first data-returning statement caused an error	SQLExecute, SQLExecDirect, or SQLParamData).
(either by a run-time error or a	Due to the
RAISERROR statement with	SQL_SUCCESS_WITH_INFO
severity greater than or equal to 11).	return code, process the results for that statement handle before it is available for use.
• There was data from any other	Expect different results as
statement, even a simple	compared to earlier versions of
RETURN statement, after the	SQL Server. Handle
error-causing statement.	SQL_SUCCESS_WITH_INFO
Due to the SQL_ERROR return code, the statement handle was available for use immediately.	using <b>SQLGetDiagRec</b> , and then call <b>SQLMoreResults</b> to process the remaining results, as appropriate.

### RIGHT (Level 2)

SQL Server 6.x	SQL Server 2000
	RIGHT is a reserved keyword and should
	not be used for database object names
	(unless using identifiers). For more
	information about SQL Server 2000 reserved
	keywords, see <u>Using Reserved Keywords</u> .
	Expect different results as compared to
	earlier versions of SQL Server because
	RIGHT is now a reserved keyword. For
	more information about using RIGHT with
	identifiers, see <u>Using Identifiers</u> .

### Security (Level 2)

SQL Server 6.x	SQL Server 2000
The GRANT and REVOKE statements granted and revoked permissions, respectively. The REVOKE statement denied a permission to a single user that was granted to the user's group.	The security model uses DENY in addition to GRANT and REVOKE. REVOKE has changed to remove a previously granted or denied permission. DENY creates an entry in the security system that denies a permission from a security account and prevents the user, group, or role from inheriting the permission through its group and role memberships. The REVOKE statement can no longer be used to deny permission to a user whose group has permission. Use the DENY statement to deny permissions explicitly to a specific user or group.
	Expect different results as compared to earlier versions of SQL Server. Recognize that scripts using the SQL Server 6. <i>x</i> security model using GRANT and REVOKE behave differently than scripts using the current model of <u>GRANT</u> , <u>REVOKE</u> , and <u>DENY</u> if REVOKE was used to deny permissions to selected members of a group.
When executing an RPC, logins using integrated security mode referred to an internal login name with the backslashes (\) translated to underscores (_). For example, \Domain\Joe was translated to	Those servers upgraded to SQL Server that execute RPC calls no longer translate backslashes to underscores when using Windows NT Authentication. To use the SQL Server version 6.x naming convention for login

Domain_Joe.	names, use <b>sp_addlinkedsrvlogin</b> to map the backslash version of the username to an underscore version.
	Expect different results as compared to earlier versions of SQL Server. Add references for <b>sp_addlinkedsrvlogin</b> to translate backslash version login names to underscore versions to maintain version 6. <i>x</i> login translations when the sending server of an RPC uses SQL Server 2000.

#### Examples

#### A. Map specific backslash login to underscore login

This example maps the \LONDON1\nancyd login name to LONDON1\_nancyd:

```
sp_addlinkedsrvlogin 'receiving_server_name',
  false,
  'LONDON1\nancyd',
  'LONDON1_nancyd', NULL
```

#### B. Map specific backslash login to sa login

This example maps Nancy's LONDON1 login to the **sa** login, because Nancy's domain login is part of the built-in administrators group:

```
sp_addlinkedsrvlogin 'receiving_server_name',
  false,
  'LONDON1\nancyd',
  'sa', NULL
```

### SELECT (Level 2)

SQL Server 6.x	SQL Server 2000
A SELECT statement without	An explicit ORDER BY clause for a
an ORDER BY clause returned	SELECT statement is required to ensure
the rows in an apparent ordered	any useful ordering of data. In addition,
set.	the exact results depend upon the collation
	being used.
	Expect different results as compared to earlier versions of SQL Server. Add an explicit ORDER BY clause to all <u>SELECT</u> statements needing to produce ordered rows.

### SET SHOWPLAN (Level 2)

SQL Server 6.x	SQL Server 2000
When SET SHOWPLAN was	When set ON, the SET SHOWPLAN_ALL
set ON, SQL Server executed	and SET SHOWPLAN_TEXT statements,
Transact-SQL statements.	which replace SET SHOWPLAN, do not
	execute Transact-SQL statements.
	Expect different results as compared to
	earlier versions of SQL Server. Expect a
	difference in behavior when <u>SET</u>
	SHOWPLAN_ALL or <u>SET</u>
	SHOWPLAN_TEXT are set to ON.

### System Tables (Level 2)

SQL Server 6.x	SQL Server 2000
System tables were used internally by SQL Server for a wide range of uses.	Some system tables have had minor changes, while others have been replaced by Information Schema Views that provide the same information.
	Expect different results as compared to earlier versions of SQL Server. Use the provided <u>Information Schema Views</u> or ODBC catalog <u>system stored procedures</u> to obtain system catalog information.
The <b>logptr</b> column of <b>sysdatabases</b> was a pointer to the transaction log.	The <b>logptr</b> column has been renamed to <b>status2</b> . Remove all references of the <b>logptr</b> column of <b>sysdatabases</b> and replace with references to the <b>status2</b> column.
The <b>dumptrdate</b> column of <b>sysdatabases</b> was the date of the last DUMP TRANSACTION.	This column is now <b>Reserved</b> . Expect different results as compared to earlier versions of SQL Server. Remove all references to the <b>dumptrdate</b> column of <u>sysdatabases</u> .
The <b>langid</b> column of <b>sysmessages</b> contained the SQL Server message group ID.	The <b>langid</b> column has been renamed to <b>msglangid</b> . Expect different results as compared to earlier versions of SQL Server. Remove all references of the <b>langid</b> column of <b>sysmessages</b> and replace with references to the <b>msglangid</b> column.
A NULL value for the <b>language</b> column of the <b>syslogins</b> table was	A NULL value for the <b>language</b> column is no longer equivalent to <b>us_english</b> .

equivalent to specifying <b>us_english</b> .	Expect different results as compared to earlier versions of SQL Server. Remove all NULL values for the <b>language</b> column of <b>syslogins</b> and replace with the name of the language to be used.
System tables obtained their column values by insertion of a specific value (SQL Server 2000 uses computed columns in many system and user-defined tables.)	System tables (and user-defined tables) can now use computed columns. Expect different results as compared to earlier versions of SQL Server. SQL Server version 6.5 queries involving table hints and system tables may still produce the same result set, but may behave differently in SQL Server 2000. For example, the query may still wait for some locks even if the NOLOCK table hint has been specified in the query's FROM clause.

### Table Hints (Level 2)

SQL Server 6.x	SQL Server 2000
These table hints (previously called optimizer hints) could be specified	Table hints must be specified following the FROM clause using a
as just the keyword following the	WITH clause. Table hints must be
FROM clause:	enclosed in parentheses.
FASTFIRSTROW,	Expect different results as compared
HOLDLOCK,	to earlier versions of SQL Server. For
INDEX,	more information, see <u>DELETE</u> ,
NOLOCK,	FROM, INSERT, SELECT, and
PAGLOCK,	UPDATE.
TABLOCK,	
TABLOCKX,	
and UPDLOCK.	

### Transactions (Level 2)

SQL Server 6.x	SQL Server 2000
When	When
CURSOR_CLOSE_ON_COMMIT	CURSOR_CLOSE_ON_COMMIT is
was set OFF, a ROLLBACK	set OFF, a ROLLBACK statement
statement did not close a Transact-	closes any Transact-SQL cursor defined
SQL cursor defined with the	with the SQL-92 form of the
DECLARE CURSOR statement.	DECLARE CURSOR statement, unless
Server cursors opened through database API functions were also	the DECLARE CURSOR statement contains either the INSENSITIVE or
left open after a ROLLBACK	STATIC keywords. All API server
statement.	cursors are also closed unless they have
	been defined as STATIC cursors (such
	as using the ODBC
	SQL_CURSOR_STATIC attribute).
	Expect different results as compared to
	earlier versions of SQL Server. Reopen
	all cursors after issuing a ROLLBACK
	statement.
The REPEATABLE READ clause	The REPEATABLE READ clause now
of the SET TRANSACTION	does not necessarily protect against
ISOLATION LEVEL statement	phantoms. Serializable transactions, set
behaved identically to the	using the SERIALIZABLE clause of
SERIALIZABLE clause. There	SET TRANSACTION ISOLATION
was no way to ensure repeatable	LEVEL, allow less concurrency than
reads without also protecting against phantoms (after a rollback,	the REPEATABLE READ clause
the value read logically never	because they protect against phantoms.
existed). Transactions that required	Expect different results as compared to
REPEATABLE READ semantics	earlier versions of SQL Server. Many
had to pay the additional	applications only need REPEATABLE
concurrency penalty of	READ semantics for correct operation.
serializability.	Use the REPEATABLE READ clause

of <u>SET TRANSACTION ISOLATION</u>
LEVEL for applications requiring
REPEATABLE READ semantics but
that do not need phantom protection. If
phantom protection is required, use the
SERIALIZABLE clause.

Here is a summary of phantom protection for both SQL Server versions 6.5 and SQL Server 2000 using SET TRANSACTION ISOLATION LEVEL.

SQL Server 6.5	SQL Server 2000
Yes	No
Yes	Yes
	Yes

### Triggers and System Stored Procedures (Level 2)

SQL Server 6.x	SQL Server 2000
<b>sp_helpsql</b> provided syntax for Transact-SQL statements, system stored procedures, and other special topics.	<b>sp_helpsql</b> is included, but no longer returns syntax information for Transact- SQL statements or system stored procedures. Executing <b>sp_helpsql</b> produces a message that recommends obtaining syntax information from Online Help. Expect different results as compared to earlier versions of SQL Server. Use SQL Server Books Online for the syntax of Transact-SQL statements and system stored procedures.
Only one trigger for each data modification event (INSERT, UPDATE, or DELETE) was allowed for each table. If a new trigger was created for a specific data modification event, it replaced the previous trigger.	<ul> <li>Microsoft® SQL Server<sup>™</sup> allows multiple triggers to be created for each data modification event (DELETE, INSERT, or UPDATE). For example, if CREATE TRIGGER FOR UPDATE is executed for a table that already has an UPDATE trigger, an additional UPDATE trigger is created.</li> <li>Expect different results as compared to earlier versions of SQL Server. Enable multiple triggers by setting the compatibility level to 80 in <b>sp_dbcmptlevel</b>. Retain SQL Server 6.<i>x</i> behavior by setting the compatibility level to 60 or 65. For more information, see <b>sp_dbcmptlevel</b> and <u>CREATE</u> <u>TRIGGER</u>.</li> </ul>

If a trigger modified the table on which it was defined, the triggers were not invoked recursively for	SQL Server allows recursive invocation of triggers. Expect different results as compared to
that modification.	earlier versions of SQL Server. Enable recursive triggers by setting the RECURSIVE_TRIGGERS database option. For more information about recursive and nested triggers, see <u>Nested</u> <u>Triggers</u> .
Several parameters of <b>sp_create_removable</b> referred to	Devices have been replaced with files and filegroups.
devices.	Expect different results as compared to earlier versions of SQL Server. Replace all device references in <b>sp_create_removable</b> with references to filegroups
References to <b>text</b> or <b>image</b> columns in either the <b>inserted</b> or <b>deleted</b> tables appeared as NULL.	References to <b>text</b> or <b>image</b> columns in both the <b>inserted</b> and <b>deleted</b> tables are no longer allowed unless the compatibility level setting of <b>sp_dbcmptlevel</b> is 60 or 65.
	Expect a difference in behavior when referring to <b>text</b> or <b>image</b> columns in <b>inserted</b> and <b>deleted</b> tables when using <u>CREATE TRIGGER</u> , depending on the setting of <u>sp_dbcmptlevel</u> .
SQL Server searched the current database followed by a search in <b>master</b> for a stored procedure using the <b>sp_</b> prefix.	Stored procedures with the prefix <b>sp_</b> are first looked up in <b>master</b> . If a user- defined stored procedure has the same name as a system-supplied stored procedure residing in <b>master</b> , SQL Server always finds the system-supplied stored procedure.

	Expect different results as compared to earlier versions of SQL Server. Expect a difference in behavior when calling user-defined stored procedures with the <b>sp_</b> prefix. Either explicitly qualify the name of the user-defined stored procedure, or rename the user-defined stored procedure.
The settings of SET ANSI_NULLS and SET QUOTED_IDENTIFIER statements were active only during the session that changed either option.	The settings of both SET QUOTED_IDENTIFIER and SET ANSI_NULLS are saved when a stored procedure is created or altered. These original settings are enabled when the stored procedure is executed, and any client session settings are restored afterward. Within the stored procedure, any changes to SET ANSI_NULLS do not take effect until after the stored procedure executes.
	Expect different results as compared to earlier versions of SQL Server. Develop databases or applications with one setting for <u>SET</u> <u>QUOTED_IDENTIFIER</u> , <u>SET</u> <u>ANSI_NULLS</u> , and all other pertinent SET options. If a client session changes SET options, do so outside of stored procedures.
When executing remote stored procedures, these procedures may have assumed non-standard behavior for the options set by SET ANSI_DEFAULTS. In addition, remote stored procedures may not have explicitly set these	When executing remote stored procedures, these procedures are executed with SET ANSI_DEFAULTS set to ON. Expect different results as compared to earlier versions of SQL Server. Expect a difference in behavior when executing

ANSI_DEFAULTS) to OFF. standa ANSI	e stored procedures if non- ard settings were used with <u>SET</u> <u>DEFAULTS</u> , or if options were splicitly set to OFF.
---------------------------------------	--

### UPDATE (Level 2)

SQL Server 6.x	SQL Server 2000
When ARITHABORT was set to	When SET ARITHABORT is OFF and
OFF, an UPDATE statement	an INSERT, UPDATE, or DELETE
encountering an arithmetic	statement encounters an arithmetic error,
overflow condition would set the	SQL Server inserts or updates a NULL
updated value to NULL, or skip	value. If the target column is not
the update if the value belonged	nullable, the insert or update action fails
to a nonnull column.	and the user receives an error.
	Expect different results as compared to earlier versions of SQL Server. Use the <i>@@ERROR</i> function to test for errors after <u>UPDATE</u> or <u>INSERT</u> statements.

## UPDATETEXT (Level 2)

SQL Server 6.x	SQL Server 2000
UPDATETEXT initialized	If the compatibility level setting of
text columns to NULL,	<pre>sp_dbcmptlevel is 65, UPDATETEXT</pre>
allocating a full 2K page.	initializes <b>text</b> columns to NULL. However,
	if the compatibility level setting is 70 or 80,
	WRITETEXT initializes <b>text</b> columns to
	NULL; UPDATETEXT initializes <b>text</b>
	columns to an empty string.
	Expect differences in behavior when
	initializing <b>text</b> values to NULL (using
	<u>UPDATETEXT</u> or <u>WRITETEXT</u> )
	depending on the compatibility level setting
	of <u>sp_dbcmptlevel</u> .

### Views (Level 2)

SQL Server 6.x	SQL Server 2000
Updatable views were restricted to modifications that affected only one table.	Updatable views can modify more than one table involved in the view. The DELETE, INSERT, and UPDATE statements can reference a view as long as SQL Server can
	translate the user's update request unambiguously to updates in the base tables referenced in the view's definition.
	Expect differences in behavior when working with updatable views with more than one table involved in the <u>DELETE</u> , <u>INSERT</u> , or <u>UPDATE</u> statements.

### Backup and Restore (Level 3)

SQL Server 6.x	SQL Server 2000
The DUMP statement created database or transaction log backups (dumps).	<ul> <li>The DUMP DATABASE and DUMP</li> <li>TRANSACTION statements are</li> <li>synonymous with BACKUP DATABASE</li> <li>and BACKUP LOG statements. Support</li> <li>for the DUMP DATABASE and DUMP</li> <li>TRANSACTION statements may be</li> <li>removed in a future release.</li> <li>Consider removing all references of</li> <li>DUMP DATABASE and replacing with</li> <li>references to BACKUP DATABASE.</li> <li>Consider removing all references of</li> <li>DUMP TRANSACTION and replacing</li> <li>with references to BACKUP LOG.</li> </ul>
The LOAD statement restored or loaded database or transaction log backups (dumps).	The LOAD DATABASE and LOAD TRANSACTION statements are synonymous with the RESTORE DATABASE and RESTORE LOG statements. Support for the LOAD DATABASE and LOAD TRANSACTION statements may be removed in a future release.
	Consider removing all references of LOAD DATABASE and replacing with references to RESTORE DATABASE. Consider removing all references of LOAD TRANSACTION and replacing with references to RESTORE LOG. For more information about RESTORE DATABASE, see <u>RESTORE</u> .

The CREATE DATABASE FOR LOAD statement syntax created a destination database	The CREATE DATABASEFOR LOAD syntax is supported for backward compatibility only. However, because SQL
before its restoration from a	Server now creates the destination
database backup and prevented	database within a restore operation, it is
anyone from using the database	recommended that the destination
between the CREATE	database not be created before executing
DATABASE, ALTER	the restore operation.
DATABASE, and LOAD	1
statements.	Do not create the database prior to restoring it.

# **Database Options (Level 3)**

SQL Server 6.x	SQL Server 2000
The <b>publish</b> option of	<b>sp_replicationdboption</b> should be
<b>sp_dboption</b> enabled or disabled	used to enable or disable publishing in
publishing in a database.	a database.
	Remove all references of <b>sp_dboption publish</b> and replace with references to <b>sp_replicationdboption</b> .

# DBCC (Level 3)

SQL Server 6.x	SQL Server 2000
DBCC NEWALLOC checked data and index pages against corresponding extent structures.	DBCC NEWALLOC is supported for backward compatibility only and is identical to DBCC CHECKALLOC.
	Consider removing all references of DBCC NEWALLOC and replacing with references to <u>DBCC</u> <u>CHECKALLOC</u> .
DBCC ROWLOCK dynamically enabled Insert Row Locking (IRL) operation on tables.	Row-level locking is automatic. DBCC ROWLOCK available for backward compatibility only. Consider removing all references of
DBCC TEXTALL selected tables in	DBCC ROWLOCK.
the database that had <b>text</b> or <b>image</b> columns and ran DBCC TEXTALLOC on them.	consistency of <b>text</b> , <b>ntext</b> , and <b>image</b> columns in a database. DBCC TEXTALL is available for backward compatibility only.
	Consider removing all references of DBCC TEXTALL and replacing with references to <u>DBCC CHECKDB</u> .
DBCC TEXTALLOC checked the allocation of <b>text</b> or <b>image</b> columns for a table.	DBCC CHECKTABLE checks the integrity of the data, index, <b>text</b> , <b>ntext</b> , and <b>image</b> pages for the specified table. DBCC TEXTALLOC is available for backward compatibility only.
	Consider removing all references of DBCC TEXTALLOC and replacing

with references to <u>DBCC</u> <u>CHECKTABLE</u> .
Use DROP DATABASE to drop or remove a SQL Server database. DBCC DBREPAIR is available for backward compatibility only. Consider removing all references of DBCC DBREPAIR and replacing with references to <u>DROP DATABASE</u> .

## **Devices (Level 3)**

The database architecture of Microsoft® SQL Server<sup>™</sup> 2000 differs from the database architecture of SQL Server 6.x. In SQL Server 2000:

- Operating system files replace database devices.
- Data files and transaction logs cannot co-exist on the same operating system file.
- A single operating system file cannot be shared by multiple databases.

For more information about database architecture, see <u>Overview of SQL Server</u> <u>Architecture</u>.

SQL Server 6.x	SQL Server 2000
DISK INIT created database or	The CREATE DATABASE statement
transaction log devices. When	syntax and ALTER DATABASE statement
DISK INIT followed either a	syntax both allow the creation of separate
CREATE DATABASE or	data and log files. Both CREATE
ALTER DATABASE statement,	DATABASE and ALTER DATABASE
SQL Server used the specified	create operating system files and databases
devices for storing the specified	in a single step (generating a log file
database or transaction log.	automatically, if none is specified with the
	LOG ON clause).
	Consider removing all references to DISK
	INIT and replacing with references to
	either CREATE DATABASE or ALTER
	DATABASE.
	DISK INIT has limited support in SOI
	DISK INIT has limited support in SQL
	Server 2000. Existing scripts will run as
	long as they do not have data and log sharing of the same data files.
	Sharing of the Same tata mes.

DISK REINIT restored device entries to appropriate system tables when the device entry was missing from <b>sysdevices</b> .	Removed; no replacement. Consider removing all references to DISK REINIT.
<b>sp_logdevice</b> put <b>syslogs</b> (contains the transaction log) on a separate database device. To add another log segment to a database with an existing log segment, it was necessary to execute DISK INIT followed by <b>sp_logdevice</b> .	Removed. The CREATE DATABASE statement creates a log file on a new operating system file. Consider removing all references to <b>sp_logdevice</b> and replacing with references to <u>CREATE DATABASE</u> . SQL Server 6.x scripts using the LOG ON clause of CREATE DATABASE will work as expected. Scripts without the LOG ON clause of CREATE DATABASE will have a log file generated automatically.
Devices created using DISK INIT and CREATE DATABASE could be dropped only by using <b>sp_dropdevice</b> .	Databases created without DISK INIT before CREATE DATABASE can be dropped with DROP DATABASE; otherwise, use <b>sp_dropdevice</b> . Use <b>sp_dropdevice</b> when using DISK INIT, followed by <u>CREATE DATABASE</u> .

## Examples

## A. Use both DISK INIT and CREATE DATABASE syntax

This example uses DISK INIT and CREATE DATABASE and works in SQL Server version 6.5 and SQL Server 2000:

DISK INIT name = 'testdb\_data', physname = 'c:\testdb\_data.dat', vdevno = 9, size = 10240

```
DISK INIT name = 'testdb_log',
physname = 'c:\testdb_log.dat',
vdevno = 8,
size = 10240
CREATE DATABASE testdb
ON testdb_data = 10
LOG ON testdb_log = 10
GO
```

# **B.** Use of sp\_logdevice and CREATE DATABASE in SQL Server 2000 fails

In earlier versions of SQL Server, this script created a 20 MB database consisting of the two files named **testdb\_data** and **testdb\_log**. This script also moved the transaction log to the **testdb\_log** device by using **sp\_logdevice**.

**Note** Scripts like this one were usually generated by the SQL Server 6.5 **sp\_help\_revdatabase** system stored procedure, which used **sp\_logdevice** to ensure the proper device layout for database restores. Because SQL Server 2000 creates the database when it is restored, scripts such as these are no longer necessary.

```
-- SQL Server 6.x example.
DISK INIT name = 'testdb_data',
    physname = 'c:\testdb_data.dat',
    vdevno = 9,
    size = 10240
DISK INIT name = 'testdb_log',
    physname = 'c:\testdb_log.dat',
    vdevno = 8,
    size = 10240
CREATE DATABASE testdb on testdb_data = 10, testdb_log = 10
-- Use sp_logdevice to move the log to the testdb_log device.
EXEC sp_logdevice testdb, testdb_log
```

In SQL Server 2000, the above script does not work the same as in SQL Server

6.x because **sp\_logdevice** no longer exists.

In SQL Server 2000, this script creates a 20 MB database consisting of the two files named **testdb\_data** and **testdb\_log**. In addition, SQL Server generates a log file automatically, which is 25 percent of the database size. In the following script (using the devices created earlier), a 10 MB log file is generated automatically:

CREATE DATABASE testdb on testdb\_data = 10, testdb\_log = 10

## C. Use CREATE DATABASE syntax only

Using the SQL Server 2000 CREATE DATABASE syntax, the database from the earlier example could be created as follows:

CREATE DATABASE testdb ON (name = 'testdb\_data', filename = 'd:\testdb\_data.dat', size = 10) LOG ON (name = 'testdb\_log', filename = 'd:\testdb\_log.dat', size = 10)

## **Open Data Services (Level 3)**

The Open Data Services gateway functions, macros, and events listed in the table are no longer supported.

Function/macro name
srv_ackattention
srv_config
srv_config_alloc
srv_errhandle
srv_event
srv_eventdata
srv_getconfig
srv_handle
srv_init
srv_langcpy
srv_langlen
srv_langptr
srv_log
srv_post_handle
srv_pre_handle
srv_run
srv_setevent
srv_terminatethread

#### **Query Performance (Level 3)**

SQL Server 6.x	SQL Server 2000
Queries could include a server user ID (SUID) without performance implications, as shown in the following table.	Queries using SUIDs continue to run and produce the same results as in earlier versions of SQL Server. However, there is a severe performance penalty because SUIDs are no longer native to the new security design. Consider removing all references to SUIDs and replacing with references to security identification numbers (SIDs) (as shown in the following table) to avoid degradation in query performance.

SQL Server 6.x SUID	Replace with SQL Server 2000 SID
SUSER_ID	SUSER_SID, which returns a SID
SUSER_NAME	SUSER_SNAME, which accepts a SID as
	input
syslogins.suid	syslogins.sid
sysdatabases.suid	<u>sysdatabases</u> .sid
sysremotelogins.suid	sysremotelogins.sid
sysusers.suid	sysusers.sid
sysalternates.suid	sysusers.isaliased
sysalternates.altsuid	sysusers.isaliased

#### Examples

#### A. Use SIDs and SUIDs to display login names of users in sysusers

This example shows SQL Server 6.*x* queries that displayed the login names of all users in **sysusers**:

```
SELECT L.name
FROM master.dbo.syslogins L, sysusers U
WHERE L.suid = U.suid
-- Or
SELECT suser_name(suid) AS name
FROM sysusers
```

Here are the queries rewritten to use SIDs rather than SUIDs:

```
SELECT L.loginname
FROM master.dbo.syslogins L, sysusers U
WHERE L.sid = U.sid
-- Or
SELECT suser_sname(sid) AS name
FROM sysusers
```

# Security (Level 3)

SQL Server 6.x	SQL Server 2000
The ON { <i>table</i>   <i>view</i> } ( <i>column</i> [, <i>n</i> ]) syntax for the GRANT statement assigned the specified permissions to the columns given for the specified table or view.	The ON { <i>table</i>   <i>view</i> } ( <i>column</i> [, <i>n</i> ]) syntax for the GRANT statement is supported for backward compatibility only. Consider using the SQL-92 standard <u>GRANT</u> syntax for object permissions and placing the column list before the ON clause.
The term integrated security allowed a SQL Server to use Windows NT Authentication mechanisms to validate logins for all connections. Standard security used SQL Server's own login validation process for all connections. Mixed security allowed login requests to be validated using either integrated or standard security.	The terms Windows Authentication and Mixed Mode replace integrated security and mixed security, respectively. Standard security no longer exists. Consider using the terms Windows Authentication and Mixed Mode rather than integrated security and mixed security. Do not refer to standard security. For more information about security modes, see <u>Authentication</u> .
The SETUSER statement allowed a database owner to impersonate another user.	SETUSER is included in Microsoft® SQL Server <sup>™</sup> 2000 for backward compatibility only, and is not recommended. This statement may no longer be supported in a future release of SQL Server. Consider removing all references to <u>SETUSER</u> .

# SELECT (Level 3)

SQL Server 6.x	SQL Server 2000
The FASTFIRSTROW optimizer hint caused the optimizer to use the nonclustered index if one matches the ORDER BY clause.	The OPTION (FAST <i>n</i> ) query hint replaces FASTFIRSTROW. However, FASTFIRSTROW is maintained for backward compatibility only. Consider removing all references to FASTFIRSTROW in <u>SELECT</u> statements and replacing with references to OPTION (FAST <i>n</i> ).
The INDEX = syntax specified one or more indexes to use for a table hint.	Supported for backward compatibility only. Consider removing all references to INDEX = and replacing (when using multiple index hints) with references to INDEX( <i>index</i> , <i>index</i> ) as shown in <u>SELECT</u> .

# SET SHOWPLAN (Level 3)

SQL Server 6.x	SQL Server 2000
The SET SHOWPLAN	The SET SHOWPLAN statement is no
statement returned output as	longer supported. It has been replaced by
informational messages	SET SHOWPLAN_TEXT and SET
through <b>SQLGetDiagRec</b> in	SHOWPLAN_ALL. The output of SET
ODBC, or through the message	SHOWPLAN_TEXT and SET
handler in DB-Library	SHOWPLAN_ALL is returned not as
applications.	informational messages, but as a result set.
	Consider removing all references of SET SHOWPLAN and replacing with references to either <u>SET</u> <u>SHOWPLAN_TEXT</u> (to display readable text) or <u>SET SHOWPLAN_ALL</u> (to display output that can be parsed more easily by an application building a report of showplan output). The application needs to process the output as part of the result set, not as messages returned through the ODBC <b>SQLGetDiagRec</b> function or the DB-Library message handler.

## System Stored Procedures (Extended) (Level 3)

SQL Server 6.x	SQL Server 2000
<b>xp_grantlogin</b> and	Use <b>sp_grantlogin</b> and <b>sp_revokelogin</b>
<b>xp_revokelogin</b> granted or	even though <b>xp_grantlogin</b> and
revoked SQL Server access to a	<b>xp_revokelogin</b> are supported for
Windows NT-based group or user.	backward compatibility only.
	Consider removing all references of <b>xp_grantlogin</b> and <b>xp_revokelogin</b> and replacing with references to <b>sp_grantlogin</b> and <b>sp_revokelogin</b> , respectively.

## System Stored Procedures (System) (Level 3)

SQL Server 6.x	SQL Server 2000
Job management was called task	The task management system stored
management, and several system	procedures are no longer documented and
stored procedures allowed	are included for backward compatibility
system administrators to create	only.
and manage tasks.	Even though Microsoft® SQL Server™
	2000 supports the task management
	system stored procedures ( <b>sp_addtask</b> ,
	sp_droptask, sp_helptask,
	<b>sp_reassigntask</b> , and <b>sp_updatetask</b> ) for
	scheduling and managing SQL Server
	jobs, consider using either SQL Server
	Enterprise Manager or the job-related
	system stored procedures listed in the
	following table for managing jobs.

The task-related stored procedures listed in the **SQL Server 6.x** column below have been replaced by the corresponding job-related stored procedures shown in the **SQL Server 2000** column.

SQL Server 6.x	SQL Server 2000
sp_addtask	sp_add_job
	sp_add_jobstep
	sp_add_jobschedule
	<u>sp_start_job</u>
sp_droptask	sp_delete_job
	<u>sp_delete_jobstep</u>
	<u>sp_delete_jobschedule</u>
sp_helptask	sp_help_jobhistory
	<u>sp_help_jobschedule</u>
	<u>sp_help_jobstep</u>
sp_reassigntask	<u>sp_purge_jobhistory</u>

	sp_stop_job
sp_updatetask	<u>sp_update_job</u>
	<u>sp_update_jobstep</u>
	sp_update_jobschedule

## Aliases (Level 4)

SQL Server 6.x	SQL Server 2000
An alias allowed a user to	Roles have replaced aliases. Because a
temporarily assume the identity	user can belong to more than one role at a
of another user within a database	time, it is no longer necessary to assume
and perform actions as the	the identity of another. Users belonging to
aliased user. For example, the	the same roles have the same permissions
database owner could be aliased	automatically, assuming permissions are
to a user so they could act as that	only applied at the role level, not the user
user, if the user were on	level.
	Expect different results as compared to earlier versions of SQL Server. Use roles instead of aliases. For more information about database roles, see <u>Managing</u> <u>Permissions</u> .

# Backup and Restore (Level 4)

SQL Server 6.x	SQL Server 2000
The LOAD HEADERONLY	The result set has changed. Expect a
statement retrieved a result	different result set from RESTORE
set detailing the header	HEADERONLY, compared to LOAD
information from a database	HEADERONLY in earlier versions of SQL
dump.	Server. For more information about
	RESTORE HEADERONLY, see <u>RESTORE</u>
	HEADERONLY.

## **Configuration (Level 4)**

Administration scripts may have used these configuration options. For more information about configuration options, see <u>sp\_configure</u> and <u>Setting</u> <u>Configuration Options</u>.

SQL Server 6.x	SQL Server 2000
for a database or transaction log dump.	<b>media retention</b> is now an advanced option. Expect different results as compared to earlier versions of Microsoft® SQL Server <sup>™</sup> . Expect the <b>media retention</b> configuration option to appear only if you have enabled the advanced configuration options of <u>sp_configure</u> .

## **CREATE PROCEDURE (Level 4)**

SQL Server 6.x	SQL Server 2000
CREATE PROCEDURE statements failed if they contained a CREATE TABLE or SELECT INTO statement creating a temporary table with the same name as a temporary table that	The CREATE PROCEDURE statement succeeds. Recode any logic that depended on the earlier behavior.
existed at the time the CREATE PROCEDURE statement was executed.	

# Data Types (Level 4)

SQL Server 6.x	SQL Server 2000
The <b>decimal</b> and <b>numeric</b> data types could use anywhere from 2 through 17 bytes to store a value, depending on the precision of the stored value.	<ul> <li>numeric and decimal now use 5, 9, 13, or 17 bytes of storage.</li> <li>Expect different results as compared to earlier versions of Microsoft® SQL Server<sup>™</sup>. Be sure that databases using the numeric or decimal data types have sufficient storage for the change in storage bytes.</li> </ul>
Results that were too small to display, called floating point underflow, returned inconsistent results for some mathematical operators and functions.	SQL Server now returns 0.0 and no error message for all instances of floating point underflow. Because of the fixed size of floating point numbers like the <b>float</b> and <b>real</b> data types, approximate numeric data have intrinsic precision and ranges of values. In cases of floating point underflow, a result of 0.0 will be returned and no error message will be displayed. For example, the mathematical calculation of 2 to the -100.0 power would have a result 0.0. Expect different results as compared to earlier versions of SQL Server. Expect different results with floating point underflow with the mathematical functions or operators. For more information, see <u>Using Mathematical</u> <u>Functions</u> . + (Add) - (Subtract)

	* (Multiply) / (Divide) ATN2 AVG CONVERT EXP POWER RADIANS SUM
A negative second parameter equal to the number of digits in the ROUND expression returned a value of 0 for <b>integer</b> , <b>float</b> , and <b>money</b> data types. When the second parameter was negative and less than the number of digits in the ROUND expression, ROUND returned a value that rounded the right-most digit down to 0.	<ul> <li>When the second parameter in the ROUND function is a negative value (for all numeric data types) that is less than the number of digits in the expression, SQL Server returns a value that is rounded up to the next digit position.</li> <li>Expect different results with the ROUND function, compared to earlier versions of SQL Server, when the second parameter is negative.</li> </ul>
The DATEADD and DATEDIFF functions returned a date value when adding or subtracting date values.	Direct date value addition and subtraction operations are supported for <u>datetime and smalldatetime</u> using the + (Add) and -(Subtract) operators. For simple date arithmetic, you can also use addition ( <u>+ (Add)</u> ) or subtraction ( <u>- (Subtract</u> )) instead of DATEADD and DATEDIFF.

## DATEPART and SET DATEFIRST (Level 4)

SQL Server 6.x	SQL Server 2000
The SET DATEFIRST setting of the DATEPART function	The <b>week</b> datepart may give values different from earlier versions of
had no effect on the <b>week</b> datepart.	Microsoft <sup>®</sup> SQL Server <sup>™</sup> . However, any difference will appear only if the SET DATEFIRST setting is not the default (the U.S. English default is 7).
	If the year provided in the DATEPART function has 366 days, a week value of 54 can be returned if the first week of the year starts on a Saturday, and the year ends on the same day of the week with the first day of the week counted from Sunday.
	When using the ISO 8601 standard, week values are always from 1 through 53, as the first week of the year is guaranteed to have a minimum of 4 days.
	Expect different results as compared to earlier versions of SQL Server. Use the default value for <u>SET DATEFIRST</u> so that <u>DATEPART</u> returns the expected results for the <b>week</b> datepart. Otherwise, DATEPART values will be one less than expected.

# DBCC (Level 4)

SQL Server 6.x	SQL Server 2000
Each DBCC statement had a certain output format.	The output formats of many DBCC statements have changed. Expect different results as compared to earlier versions of Microsoft® SQL Server <sup>™</sup> .
The DBCC PERFMON and DBCC SQLPERF statements documented SQL Server performance statistics used for studying SQL Server performance.	No longer documented. These statements may change in a future release of SQL Server. Use the Windows 2000 System Monitor Windows NT 4.0 Performance Monitor to monitor the performance counters for SQL Server. For more information, see <u>Monitoring with Windows Performance</u> <u>Monitor</u> .

# **DBCS String Comparisons (Level 4)**

SQL Server 6.x	SQL Server 2000
When comparing DBCS space characters, the Unicode A140 space character (U-A140) was not equal to the Unicode 0020 (U-0020) space character.	Comparisons involving the Unicode A140 space character (U-A140) are now equivalent to the Unicode 0020 (U-0020) space character. Expect different results as compared to earlier versions of SQL Server when comparing DBCS space characters.

# **DELETE and SELECT (Level 4)**

## **Devices (Level 4)**

SQL Server 6.x	SQL Server 2000
The DISK RESIZE statement	The DISK RESIZE statement is
altered the size of a database	supported, but may not be supported in
device.	future releases. In addition, the DISK
	RESIZE statement does not alter the size
	of the database. Instead, use ALTER
	DATABASE.
	Expect different results as compared to earlier versions of Microsoft® SQL
	Server <sup>™</sup> . Use the MODIFY FILE clause
	of the <u>ALTER DATABASE</u> statement to
	alter the size of a database.

# Functions (Level 4)

SQL Server 6.x	SQL Server 2000
The @@DBTS global variable	The value returned by the @@DBTS
was incremented any time any	function changes only if a row
page in the database was modified	containing a <b>timestamp</b> column is
in any way.	modified.
	Expect different results as compared to earlier versions of Microsoft® SQL Server <sup>™</sup> when using <u>@@DBTS</u> .

## **Global Variables (Level 4)**

Pre-SQL Server 7.0	SQL Server 7.0
Global variables were system-supplied,	
1	a form of function and are now referred to as functions.
having two at symbols ( $@@$ ) preceding their names.	For more information, see <u>Functions</u> .

## ODBC (Level 4)

SQL Server 6.x	SQL Server 2000
In the version 2.65 ODBC driver, the long- running query interval, specified by calling <b>SQLSetConnectOption</b> with the driver- specific connection option SQL_COPT_SS_PERF_QUERY_INTERVAL, was specified in seconds.	The SQL_COPT_SS_PERF_QUERY value is specified in milliseconds. Expect different results as compa- versions of Microsoft® SQL Serv Multiply the value of SQL_COPT_SS_PERF_QUERY by 1,000 to convert the number o milliseconds. For more informatic SQL_COPT_SS_PERF_QUERY see <u>SQLSetConnectAttr</u> .
For earlier versions of the ODBC SQL Server driver, messages from consecutive PRINT, RAISERROR, DBCC, or similar statements (in a batch or stored procedure) were combined into a single result set.	For the ODBC SQL Server 3.51- driver (included with SQL Server messages from consecutive PRIN RAISERROR, DBCC, or similar (in a batch or stored procedure) a a separate result set for each state Expect different results as compa- versions of SQL Server. Call <b>SQLMoreResults</b> to process the from each statement.
Earlier versions of the ODBC SQL Server driver returned SQL_SUCCESS when executing a searched UPDATE or DELETE statement that affects no rows (using <b>SQLExecute, SQLExecDirect,</b> or <b>SQLParamData</b> ). <b>SQLRowCount</b> returned zero.	When an ODBC version 3. <i>x</i> appli the ODBC SQL Server 3.5 driver with this release, it returns SQL_] when executing a searched UPDA DELETE statement that affects no <b>SQLExecute</b> , <b>SQLExecDirect</b> , <b>c</b> <b>SQLParamData</b> ). <b>SQLRowCou</b> returns zero.

Expect universities as compa
versions of SQL Server. Handle
SQL_NO_DATA appropriately.

# **Rebuilding the master Database (Level 4)**

SQL Server 6.x	SQL Server 2000
rebuilt the <b>master</b> database.	No longer supported. SQL Server includes the Rebuild Master ( <b>rebuildm</b> ) utility. Use the <u>Rebuild Master (<b>rebuildm</b>)</u> <u>Utility</u> located in the x:\Program Files\Microsoft SQL Server\80\Tools\Binn folder to rebuild the <b>master</b> database.

## **Rebuilding the Registry (Level 4)**

**Note** In Microsoft<sup>®</sup> SQL Server<sup>™</sup> 2000, this utility is replaced by the setup option, Registry Rebuild.

Pre-SQL Server 7.0	SQL Server 7.0
-	No longer supported. Instead, use
statement rebuilt the registry:	Setup to rebuild the registry.
setup/t RegistryRebuild = On.	

## **Replication (Level 4)**

SQL Server 6.x	SQL Server 2000
Subscriptions to one or more articles in a publication were created either through SQL Server Enterprise Manager or through the appropriate system stored procedures.	SQL Server Enterprise Manager no longer allows subscription to one or more articles. Subscribing to one or more articles of a publication can be done only by using the appropriate replication system stored procedures.
	Expect different results as compared to earlier versions of SQL Server. Use the replication system stored procedures to subscribe to one or more articles of a publication. When using SQL Server Enterprise Manager, it is necessary to subscribe to the entire publication.

Replication functions differently between SQL Server 2000 and SQL Server version 6.5 servers. In addition, SQL Server 2000 offers enhanced scripting ability after your replication topology is created in the user interface. This enhanced scripting allows mass implementation of replication topology with a minimum of time and effort.

# Security (Level 4)

SQL Server 6.x	SQL Server 2000
The SYSTEM_USER niladic function returned nulls for any	The appropriate domain and login names are returned if Windows
Microsoft Windows NT® login.	Authentication is used with the SYSTEM_USER function.
	Expect a different result, as compared to earlier versions, when using <u>SYSTEM_USER</u> with Windows Authentication.

# SELECT (Level 4)

SQL Server 6.x SO	QL Server 2000
right outer join (=*) operators Or were used in SELECT is statements to produce left and su right outer joins, respectively. M It ref rig SI ref sta an of	The SQL-92-standard syntax of LEFT OUTER JOIN and RIGHT OUTER JOIN is preferred. However, join operators upported in earlier versions of Aicrosoft® SQL Server <sup>™</sup> are supported. is recommended that you remove all eferences of the left outer join (*=) and ght outer join (=*) operators in all ELECT statement FROM clauses and eplace with references to the SQL-92- tandard syntax RIGHT OUTER JOIN and LEFT OUTER JOIN. Future versions of SQL Server will support only the SQL- 2-standard syntax.

# Triggers and System Stored Procedures (System) (Level 4)

SQL Server 6.x	SQL Server 2000
<b>SQL Server 6.x</b> Returned values were not always correct for <b>text</b> or <b>image</b> columns in either the <b>inserted</b> or <b>deleted</b> tables when either table was used in a CREATE TRIGGER statement.	NULL values are returned for <b>text</b> or <b>image</b> column references in the <b>inserted</b> or <b>deleted</b> tables in CREATE TRIGGER. Expect different results as compared to earlier versions of SQL Server. Remove all references to either the <b>text</b> or <b>image</b>
	columns of the <b>inserted</b> or <b>deleted</b> tables when used in <u>CREATE</u> <u>TRIGGER</u> statements.
Direct recursion of triggers (the ability of a trigger to call itself) was not supported, but indirect recursion was allowed.	Direct trigger recursion is enabled with the RECURSIVE_TRIGGERS option of ALTER DATABASE. Indirect recursion is enabled with the <b>nested triggers</b> configuration option. Expect different results as compared to earlier versions of SQL Server.
Server-to-server communication existed between version 4. <i>x</i> and version 6. <i>x</i> servers when initialized by either side.	Version 4. <i>x</i> or 6. <i>x</i> servers can communicate with SQL Server 2000 servers. However, server-to-server communication is not supported from SQL Server 2000 servers to 4. <i>x</i> servers. Expect different results as compared to earlier versions of SQL Server. Upgrade the 4. <i>x</i> server to either SQL Server 6. <i>x</i> or SQL Server 2000
<b>sp_tableoption</b> set option values for user-defined tables, including	The Insert Row Locking (IRL) parameters in <b>sp_tableoption</b> are not

the use of Insert Row Locking (IRL).	supported but have been replaced with complete row-level locking. Expect different results as compared to earlier versions of SQL Server. Remove all references to IRL actions implemented using <u>sp_tableoption</u> and use the built-in row-level locking of SQL Server 2000 instead. Applications calling <b>sp_tableoption</b> should continue to work properly; the IRL parameters will be ignored.
The <b>@message</b> parameter of <b>xp_readmail</b> was <b>varchar(255)</b> . The <b>@message</b> and <b>@query</b> parameters of <b>xp_sendmail</b> were <b>varchar(255)</b> .	The <b>@message</b> and <b>@query</b> parameters are now <b>varchar(8000)</b> . Expect differences in behavior as compared to earlier versions of SQL Server when using the <b>@message</b> parameter of <b>xp_readmail</b> and the <b>@message</b> and <b>@query</b> parameters of <b>xp_sendmail</b> .

## UPDATE (Level 4)

SQL Server 6.x	SQL Server 2000
CREATE TABLE t1 (c1 int) GO INSERT t1 VALUES (1) INSERT t1 VALUES (2) GO	Syntax no longer supported UPDATE keyword. The UF UPDATE a1 SET c1 = 50 FROM t1 a1, t1 a2 WHERE a1.c1 = 1 ANI a2.c1 = 2 Expect differences in behav
This UPDATE statement with table and alias references worked. USE pubs GO UPDATE titles SET t.ytd_sales = t.ytd_sales + s.qty FROM titles t. sales s	The alias specified after the following the SET keyworc must be changed to 65 for t 6.x. Here is the same UPDATE USE pubs GO UPDATE t SET t.ytd_sales = t.yt FROM titles t, sale WHERE t.title_i AND s.ord_date GO

Expect differences in behav
different table references fc
statement and the SET keys

#### **Utilities (Level 4)**

SQL Server 6.x	SQL Server 2000
<b>isql/w</b> used DB-Library. The SQL-92 settings, like ANSI_WARNINGS, were set off, by default.	SQL Query Analyzer uses the SQL Server ODBC driver, which, by default, sets these SQL-92 options on: SET ANSI_WARNINGS, SET ANSI_PADDING, and SET ANSI_NULLS. Any errors returned are formatted as ODBC errors rather than DB-Library errors. Expect different results as compared to earlier versions of Microsoft® SQL Server <sup>™</sup> . Expect different results with SQL Query Analyzer, compared to the <b>isql</b> utility or the SQL Server 6.5 version of <b>isql/w</b> .
<b>isql/w</b> used the ANSI-ISO code pages. When connected to a server using OEM code page 850 or 437, <b>ANSI to OEM character</b> <b>translation</b> had to be explicitly enabled. Otherwise, data with extended characters appeared garbled.	The SQL Server 2000 ODBC driver automatically detects the need for and sets up automatic ANSI to OEM conversion. In addition, when SQL Query Analyzer connects, automatic detection is enabled. Expect a change in behavior when using SQL Query Analyzer with international or extended characters.

For additional information about changes to SQL Query Analyzer, see the discussion "Default Connection Option Settings in SQL Query Analyzer" in <u>SQL Server 2000 and SQL Server version 7.0</u>.