SQL Console Guide

20-2100-1004

Trademarks

Copyright

SQLConsole Guide
20-2100-1004
November 1997

Contents

Preface

The *SQLConsole Guide* describes how to use SQLConsole to simplify database administration. In this preface, you find information about:

- Who should read this manual.
- What is in this manual.
- Conventions used in the manual.
- Other helpful resources.
- How to send us your comments.

Audience

The *SQLConsole Guide* is written for anyone who needs to perform database administration tasks with SQLBase servers and databases. This includes:

- **Database Administrators** (DBAs) who want to simplify or automate database administration activities.
- **Programmers** who want to monitor system performance, create and manage database objects such as user accounts and stored procedures, or perform maintenance tasks such as database backup and recovery.
- **Other users** who want a graphical interface for creating and using SQLBase database objects.

This manual assumes you are familiar with the basics of Windows such as opening, closing, and resizing windows, and clicking and dragging. We also assume that you understand basic SQLBase concepts.

What is in this manual

The *SQLConsole Guide* describes how to use SQLConsole to simplify database administration. It shows how to use SQLConsole to manage database objects, monitor performance, and automate database maintenance.

Chapter 1 provides an overview of the work you can perform with SQLConsole.

Chapter 2 provides installation and configuration information.

Chapter 3 shows how to get started with SQLConsole. You learn how to use the SQLConsole interface and how to perform basic tasks such as examining objects in a database.

Chapters 4 through *11* provide more detailed information on using various SQLConsole components. You can refer to the chapters that apply to the tasks you want to perform with SQLConsole.

An appendix lists the various server and database properties you can set with SQLConsole.

The manual concludes with a glossary and index.

Notation conventions

Before you start using this manual, it is important to understand the typographical conventions we use in this manual:

Formatting Convention	Type of Information
You	A developer who reads this manual
User	The end-user of applications that you write
bold type	Menu items, push buttons, and field names. Things that you select. Keyboard keys that you press.
Courier 9	Builder or C language code example
SQL.INI MAPDLL.EXE	Program names and file names
Warning:	Precaution
Important:	Vital information
Note:	Supplemental information
Alt+1	A plus sign between key names means to press and hold down the first key while you press the second key

Other helpful resources

Centura Books Online. The Centura document suite is available online. This document collection lets you perform full-text indexed searches across the entire document suite, navigate the table of contents using the expandable/collapsible browser, or print any chapter. Open the collection by selecting the Centura Books Online icon from the **Start** menu or by double-clicking on the launcher icon in the program group.

Online Help. This is an extensive context-sensitive online help system. The online help offers a quick way to find information on topics including menu items, functions, messages, and objects.

World Wide Web. Centura Software's World Wide Web site contains information about Centura Software Corporation's partners, products, sales, support, training, and users. The URL is http://www.centurasoft.com.

To access Centura technical services on the Web, go to http:/www.centurasoft.com/support. This section of our Web site is a valuable resource for customers with technical support issues, and addresses a variety of topics and services, including technical support case status, commonly asked questions, access to Centura's Online

Newsgroups, links to Shareware tools, product bulletins, white papers, and downloadable product updates.

For information on training, including course descriptions, class schedules, and Certified Training Partners, go to http://www.centurasoft.com/training.

Send comments to...

Anyone reading this manual can contribute to it. If you have any comments or suggestions, please send them to:

Technical Publications Department
Centura Software Corporation
975 Island Drive
Redwood Shores, CA 94065

or send email, with comments or suggestions to:

techpubs@centurasoft.com

Chapter 1

Introduction to SQLConsole

This chapter introduces you to SQLConsole. It contains an overview of SQLConsole's features and tells you where to go for more information.

SQLConsole Overview

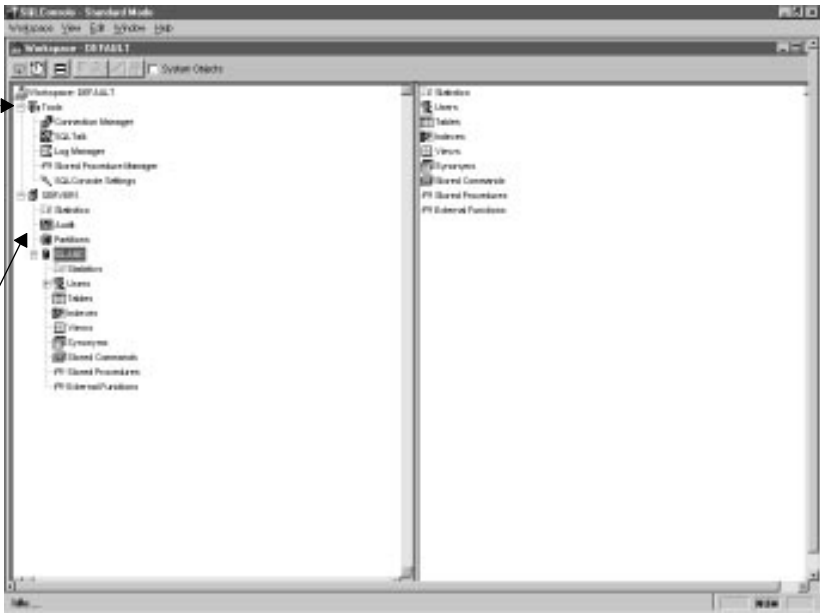
SQLConsole is a graphical database administration and monitoring tool for SQLBase servers and databases. From a single Windows desktop, SQLConsole lets you easily perform administrative tasks for a local SQLBase server and all the SQLBase servers on a network.

Simplify database administration

SQLConsole shows database objects graphically. This makes it easy to view or make changes to your tables, user accounts, stored commands, external functions, and other database objects. The following picture shows you a SQLConsole window.

The Tools node lets you access database administration

The Server node and the child nodes below it let you access database objects and statistical information.



Automate common database tasks

SQLConsole's Scheduling Manager automatically executes many DBA functions—such as initiating a backup, checking database integrity, and updating optimizer statistics—and logs the results of automated operations for later verification.

The Scheduling Manager lets you schedule data- and time-intensive operations when they cause the least disruption, reducing system management costs and improving network performance.

Monitor database performance

To help you tune your system, you can use SQLConsole's statistics windows. The statistics windows are a valuable tool for performance tuning because they provide an objective measure of how your system performs over time and under typical conditions. You can view server and database activity down to the process, cursor, and lock level.

You can use this detailed information, presented in easy-to-read forms and online graphs, to make sound optimization decisions and detect problems.

Detect potential trouble

To help you circumvent potential problems, SQLConsole's Alarm Manager alerts you when user-defined thresholds for various server conditions are exceeded. The alarm can trigger a predefined action, such as killing the connection that is causing the alarm condition.

And more...

You can use SQLConsole's graphical interface to perform DBA operations, like database load and unload operations, without the need for SQL commands. You can easily change server or database properties. And you can also audit a SQLBase server, for example, monitor who logs on to a database and which tables they access.

Where to go from here

To install and start SQLConsole, refer to the online installation instructions and *Chapter 2*. After you start SQLConsole, use the following table as a guide for your next steps.

If You Want To...	Read this Chapter
Begin using SQLConsole by opening and using a workspace. This is the foundation for almost any activity you perform with SQLConsole.	Chapter 3
Customize a workspace.	Chapter 4
Perform server or database operations.	Chapter 5
Monitor your system's performance.	Chapter 7
Perform server auditing.	Chapter 8
Set up automated responses to certain server conditions.	Chapter 9

If You Want To...	Read this Chapter
Automate database maintenance tasks.	Chapter 10
Manage user accounts and information.	Chapter 11

Chapter 2

Requirements and Startup

This chapter discusses:

- The SQLConsole environment
- System requirements
- Installing SQLConsole
- Startup options
- Customizing SQLConsole settings.

The SQLConsole environment

SQLConsole is available on these platforms:

- Windows 95
- Windows NT

You can run SQLConsole locally (where SQLConsole and all SQLBase files reside on a local machine), in a LAN environment (where the client and server programs run on different machines), or both. Refer to CERTIFY.TXT for detailed information about supported platforms and configurations.

System requirements

To use SQLConsole, you need the hardware and software listed in the following table.

Category	Requirements
Computer	An IBM 386-compatible or higher running either Windows NT or Windows 95.
Monitor	VGA capabilities
Memory	8 MB minimum; 16 MB recommended
Disk space	25 MB minimum
SQLBase	SQLBase version 6.0 or higher.

Installing SQLConsole

SQLConsole is installed with SQLBase. To install SQLConsole, follow the instructions that appear on your screen when you install SQLBase.

Startup options

When you start SQLConsole for the first time, it asks you to choose a startup option. The following sections describe each option.

Running in standard mode

During startup, SQLConsole asks if you want to disable SQLConsole's automated features. If you say yes, SQLConsole enables *standard mode*. Standard mode is an option that speeds application startup but disables SQLConsole's automated features

(enhanced mode). In standard mode, the title bar displays "standard mode." If you run in standard mode, you do not need to install the SQLCON database.

Other than at startup, you can enable standard mode:

- With the **General** tab of the SQLConsole Settings property sheet. For more information, read *SQLConsole Settings* on page *5-8*.
- By starting SQLConsole with a command line argument:

```
SQLCON61.EXE -T
```

Running in enhanced mode

Enhanced mode enables the following automated features:

- Alarm Manager, which can alert you when user-defined thresholds for various server conditions are exceeded.
- Scheduling Manager, which lets you automate database maintenance activities.
- Mail Manager, which lets you use e-mail with SQLConsole.
- Calendar, which lets you view maintenance events that you schedule with the Scheduling Manager.
- Logging database statistics to the SQLCON database.

Use standard mode if you do not require these automated features.

In enhanced mode, the title bar displays "enhanced mode."

The SQLCON database

If you start SQLConsole in enhanced mode, SQLConsole looks for a SQLCON database. If the database does not exist, it asks if you want to create it. The SQLCON database performs several functions:

- Stores the settings for the Alarm Manager.
- Stores the settings for the Scheduling Manager.
- Stores logging information from the server monitoring and information windows.
- Acts as an outbox for Mail Manager (e-mail) support.

If you want to use any of these features, you must install a SQLCON database on a local database server. This means you need to install SQLBase server software on the machine where you want to install the SQLCON database if the server software is not already installed there.

If the SQLCON database is on a networked server, each copy of SQLConsole writes data to the single SQLCON database. We do not recommend this configuration because the SQLCON database does not handle input from multiple copies of SQLConsole.

If you remove or prevent the installation of the SQLCON database, Alarm Manager, Scheduling Manager, Mail Manager, and server Event Manager logs are disabled.

Note: If you have a SQLBase version 6.0 SQLCON database (from a SQLConsole version 6.0 release), SQLConsole automatically alters it so it runs against version 6.1 and higher.

Customizing SQLConsole settings

Depending on your requirements, you may want to change certain default settings. You do this with the SQLConsole Settings node (under the Tools node) on a workspace.

You can change a setting at any time. When you change a setting, the new setting takes effect immediately. The new setting is saved when you close the session, so subsequent SQLConsole sessions also use the new setting. SQLConsole Settings apply to *all* workspaces.

In general, you may want to change one or more SQLConsole settings if you want to:

- Change the rate at which the SQLConsole activity screens refresh.
- Switch between standard and enhanced mode.
- Log information about server activity (see *Logging Information Statistics* in Chapter 7).
- Use SQLConsole with your e-mail system.
- Configure SQLConsole to work with a pager.

To change a SQLConsole setting, you must first open a workspace (as described in Chapter 3) and then use the procedure *Change a SQLConsole setting* on page *5-8*.

Chapter 3

Getting Started: Using a Workspace

Nearly every action you perform in SQLConsole is performed on a workspace. This chapter explains:

- What a workspace is.
- How to use the workspace interface.
- How to open a workspace.
- How to use a workspace to create, alter, and drop database objects.

What is a SQLConsole workspace?

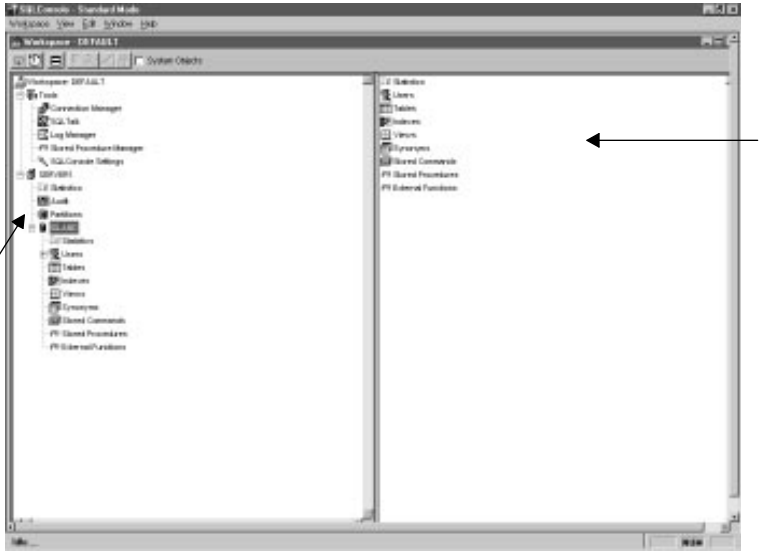
When you work with SQLConsole, you perform actions on a SQLConsole *workspace*. Typically, a workspace shows:

- One or more server nodes, which show a graphical representation of your servers and databases.
- A Tools node, which shows various SQLConsole database administration tools.

A workspace is a combination of two windows (also called *panes*). The window on the left is the main window. It shows an overall view of the tools and servers and is where you click to perform your work. The window on the right shows you a detailed view of the node you select in the left window.

This picture shows you a sample workspace.

A server node. The child nodes under it represent performance statistics, databases, and other items.



This window shows detailed information about the node you select in the window on the left.

A workspace organizes your SQLConsole tools, servers, and databases. You can build several workspaces, each customized for certain tasks. For example, if you do not want certain databases to appear on a workspace, you can build a customized workspace that hides those databases.

A workspace makes a SQLBase server easier to understand and manage by representing each part of a server as a node (or *branch*) in a tree structure, much like Microsoft's and Explorer does for files and directories.

Note: If you run in standard mode, certain tools do not appear on your workspace. For more information, read *Running in standard mode* on page *2-2*.

Which servers and databases appear on a workspace?

You must manually connect to servers and databases using SQLConsole's Connection Manager. For information on using the Connection Manager, read *Chapter 6, DBA Operations.* If the connection is successful, a node representing the server appears on any workspace you open (unless you customize the workspace to prevent the server node from appearing, as described in *Chapter 4*).

When you establish a server connection, you can also establish a connection to all the installed databases on that server, and a node for each database appears on the workspace.

Using a workspace

SQLConsole is based on the Windows 95 interface, and you navigate as you would a Windows 95 application.

Right-click to see a context menu

Many SQLConsole tasks require that you right-click a workspace node to display its context menu. The context menu shows which actions you can perform on the node: enabled actions are black, and disabled actions are grey.

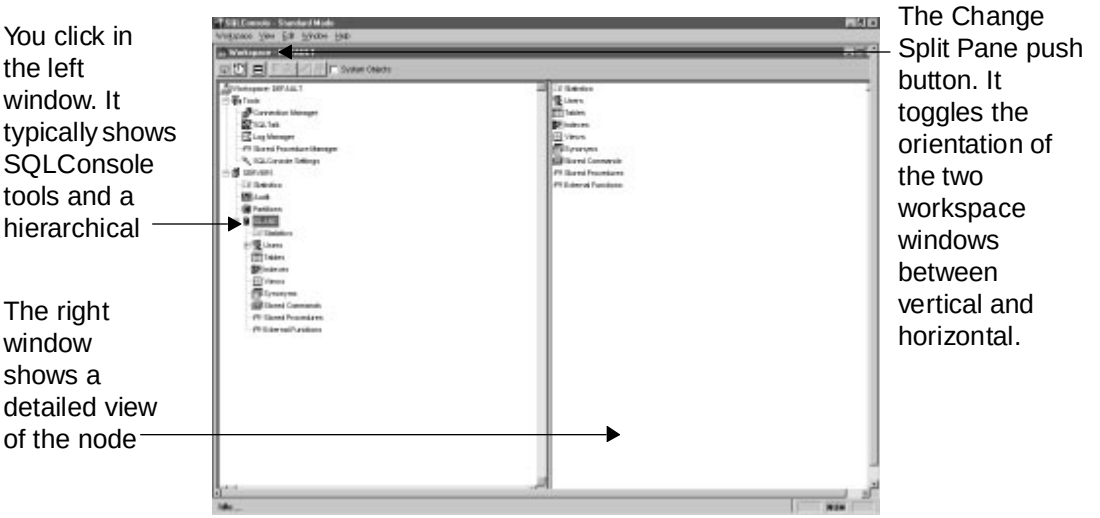
Open and close a node

Double-clicking a node in the left workspace window expands the node to display the child nodes underneath it. If the node you select has no child nodes, then double-clicking the node opens a dialog box or information window. Double-clicking an expanded node closes it.

Alternatively, you can:

- Right-click a node and then select **Explore** from the context menu. This expands the node. Selecting **Explore** on an expanded node closes the node.
- Use the **Plus** (+) and **Minus** (-) keyboard keys to expand and contract a node.

As you expand nodes in the left window, the right window changes to display either child nodes or details about the selected node.



SQLConsole provides a default workspace

SQLConsole provides a special workspace named DEFAULT. The DEFAULT workspace contains all possible tools and nodes that can appear on a workspace. It is the superset of all workspaces.

When you begin using SQLConsole, your first step is to open the DEFAULT workspace. Later, you can use the DEFAULT workspace to create your own customized workspaces.

Opening a workspace

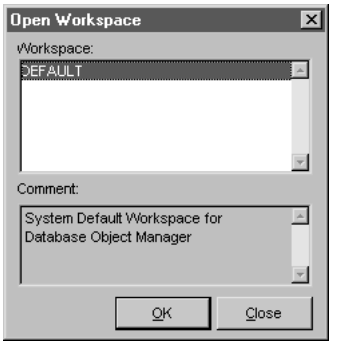
This section describes how to open a workspace. When you begin using SQLConsole, start by opening the DEFAULT workspace. (If you want, you can make a customized workspace later, as described in *Chapter 4*.) The DEFAULT workspace opens automatically.

This procedures assumes that you have opened SQLConsole by clicking the SQLConsole icon in the SQLBase program group. If you are starting SQLConsole for the first time, SQLConsole asks you a series of questions. For more information, read *Startup options* on page *2-2*.

Open a workspace

1. Select **Workspace**, **Open**

   The Select Workspace dialog box opens. It lists the workspaces automatically provided by SQLConsole (DEFAULT) and any other workspace you created.



2. Click on the workspace you want to open and click **OK**. If you have not yet created a workspace, select the DEFAULT workspace.

   SQLConsole displays the workspace. For information on navigating and opening the nodes on the workspace, read *Using a workspace* on page *3-3*.

Note: You can open multiple instances of a workspace, if required.

Creating, altering, or dropping database objects

A workspace's graphical interface makes it easy to create, alter, or drop items in a server or database.

Create, alter, or drop a database object

1. Right-click a node in a left workspace window to display its context menu.

   For example, right-click the Tables node (under a database node).

2. From the context menu, select the action you want to perform.
3. Enter information in the dialog box that appears.
4. Click **OK**.

Alternatively, you can click a database object node and then click either the **Create**, **Alter**, or **Drop** push button (assuming the push button is enabled). Whenever you click a node, the push buttons enable and disable themselves to show you the valid actions for that particular node.

Note: To identify a push button, position your cursor over it and wait a few seconds for its label to appear. For example, when you put your cursor over the push button with a hammer, a "create" label appears.

For more information on viewing the nodes on the workspace, read *Using a workspace* on page *3-3*.

The View menu

This section describes the **View** menu options. To turn on any of these options, open a workspace, select **View**, then click the option (so a check appears next to it). To turn off an option, select **View** and then click the option to make the check disappear.

Show System Objects

With a workspace open, select this option from either the **View** menu or the System Objects check box on the workspace tool bar. When you turn on this option, the workspace displays both system catalog objects (for example, the SYSTABLES and SYSINDEXES tables) and user-defined objects.

Using this option increases SQLConsole's overhead and thus slows performance. Once you are accustomed to the SQLConsole interface, we recommend that you turn off this option.

Show Qualified Names

When this option is enabled, SQLConsole displays fully-qualified object names where appropriate. For example, the fully-qualified name for table SYSTABLES is SYSADM.SYSTABLES. This setting has no performance impact.

Performance issues

Use the System Objects option (described above) only if needed since it can slow SQLConsole performance.

In addition, applications that perform UPDATE, CREATE, or DROP commands and that fail to commit these transactions will cause SQLConsole to time out when it tries to select information from the database. There are two solutions to this problem:

- Preferably, locate the application that creates the exclusive lock and make it commit its transactions. Use the Lock statistics window (under the Statistics node for each database) to search for the offending application.
- If you cannot find the application that creates the exclusive lock, you can set the database to enable read-only mode and then reconnect SQLConsole to the

database. SQLConsole detects that read-only mode is enabled and connects a read-only isolation level cursor for selecting its information, thus eliminating the timeout problem.

Be aware that enabling read-only mode for a database has negative performance implications and should be considered only a temporary fix.

Note: To see how to enable read-only mode through SQLConsole, read *Appendix A, Server and Database Properties*.

Function key shortcuts

You can use the function keys to perform the following tasks:

- **F5** — Refresh current window.
- **F6** — Refresh all windows.

Chapter 4

The Workspace Designer

You can perform all your work with the DEFAULT workspace that SQLConsole provides, or you can customize a workspace so only the nodes you want to see are available on it. This chapter shows how to use the Workspace Designer to customize a workspace.

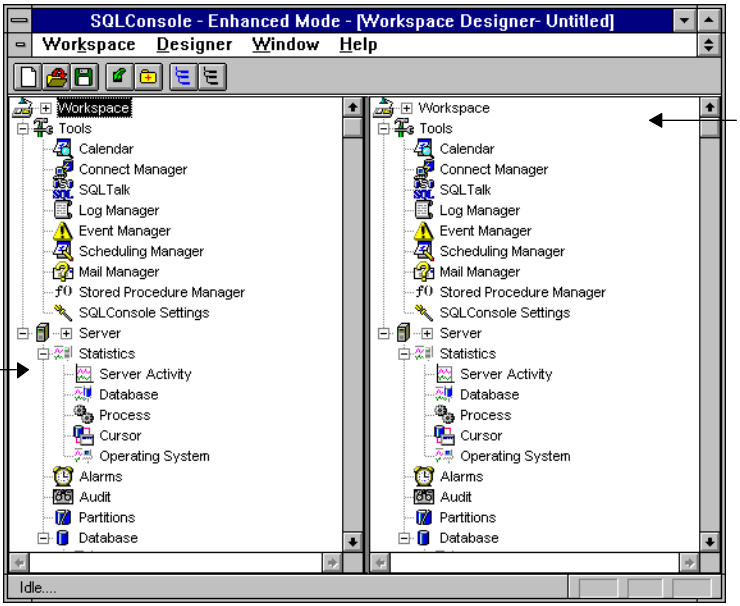
What is the Workspace Designer?

The Workspace Designer is a workspace editor for altering which nodes are available on a workspace. You can use it to alter any workspace except the DEFAULT workspace that SQLConsole provides. For example, you can create a workspace that shows a subset of:

- Tools. For example, do not show the Stored Procedure Manager node.
- Database object nodes. For example, do not show the Stored Commands, Stored Procedures, or External Functions nodes.

The following picture shows the Workspace Designer.

This window is the workspace template. It displays all possible nodes that can appear on a



This window shows you only the nodes that will appear on the workspace you are designing. For example, we turned off the SQLTalk node (under

The Workspace Designer windows

In the Workspace Designer, the window on the left is your template. It shows you all possible nodes you can include on a workspace. The window on the right shows you only the nodes that will be available on the workspace after you leave the Workspace Designer and then open the workspace.

If you have a color monitor, the black nodes in the left window are enabled. This means they *will* be available on the workspace you are designing. The blue nodes are disabled and will *not* be available on the workspace you are designing.

Creating a new workspace

Although you can use the DEFAULT workspace for all your SQLConsole work, you may find it useful to create a new workspace that you customize according to your environment.

Create a new workspace

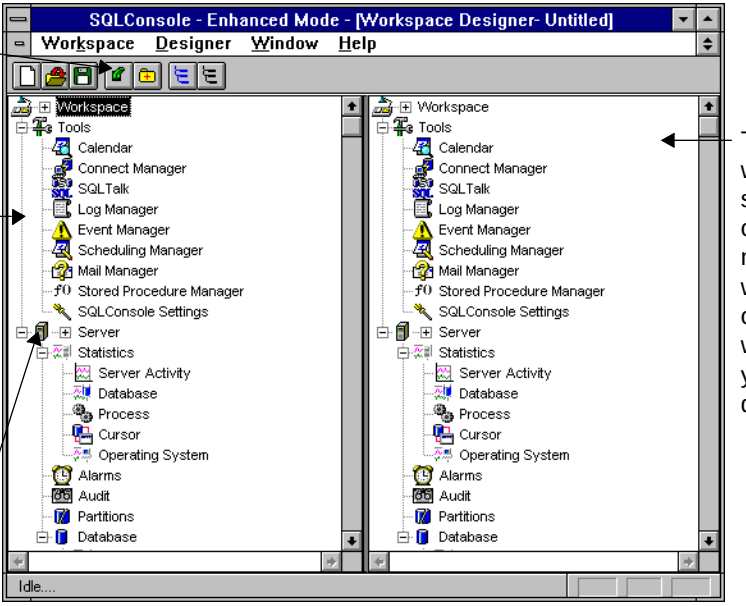
1. Select **Workspace**, **Design New**.
2. In the dialog box that appears, select the template you want to use for the new workspace (DEFAULT, for example), then click **OK**.

   The Workspace Designer appears.

The Enable push button

This window is the template. The Enable push button sets whether a node appears on the new workspace.

The template shows a generic Server node (containing a Database child node). The settings specified for the Server node apply to all Servers that appear when you exit the Workspace Designer and open the new workspace.



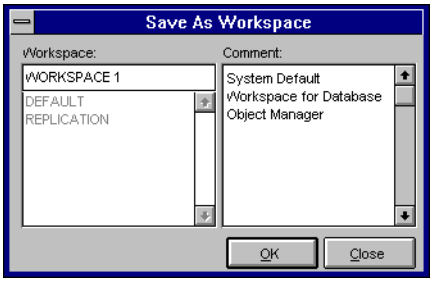
This window shows you only the nodes that will appear on the workspace you are designing.

In the Workspace Designer, the left window is your template and shows all possible nodes that can appear on the workspace you are designing. The right window shows only the nodes that will appear on the workspace (after you exit the Workspace Designer and then open the workspace you are now designing).

3. Customize the new workspace as needed. Use one or more of the methods described in the Customizing a workspace section that follows to:
   - Make a node appear on a workspace.
   - Remove a node from a workspace.

- Set which servers or databases appear on a workspace.
- Set which database objects appear on a workspace.

4. Select **Workspace**, **Save As**.
5. In the **Save As Workspace** dialog box, enter the new workspace name. You can enter up to 25 alphanumeric characters. (Workspace names are stored in the SQLCON.INI file.)



6. If you want, enter comments in the Comment field.
7. Click **OK**.
8. Close the Workspace Designer window.

   You can now open the new workspace by selecting **Workspace**, **Open**.

Altering an existing workspace

Altering a workspace lets you change any of the following:

- Which tools nodes appear on the workspace. For example, you can change whether the SQLTalk node (under the Tools node) appears on a workspace.
- Which database objects and statistics nodes appear on a workspace.
- Which servers or databases appear on a workspace.

You cannot alter the workspaces that SQLConsole provides (DEFAULT).

Alter a workspace

You can alter any workspace that you create, but you cannot alter the DEFAULT workspace.

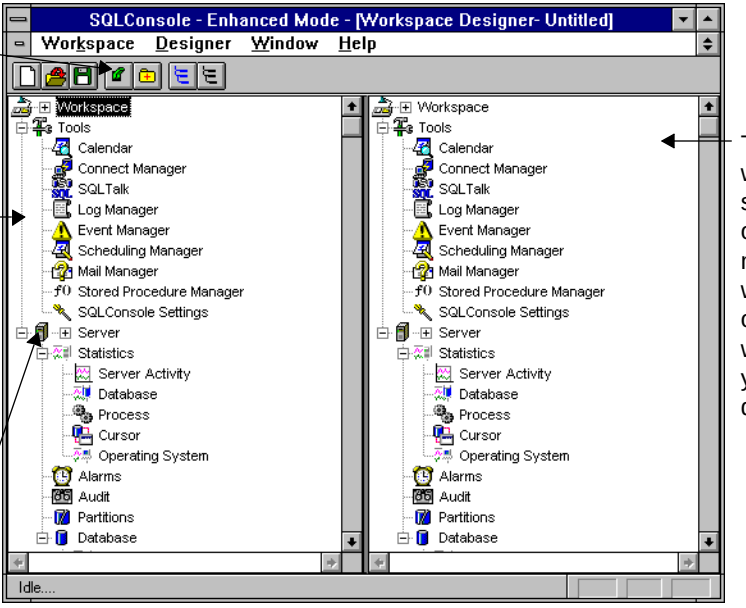
1. Select **Workspace**, **Open**.
2. Select the workspace you want to alter and click **OK**.
3. Select **Workspace**, **Alter**.

The Workspace Designer appears.

The Enable push button

This window is the template. The Enable push button sets whether a node appears on the new workspace.

The template shows a generic Server node (containing a Database child node). The settings specified for the Server node apply to all Servers that appear when you exit the Workspace Designer and open the new workspace.



This window shows you only the nodes that will appear on the workspace you are designing.

In the Workspace Designer, the left window is your template and shows all possible nodes that can appear on the workspace you are designing. The right window shows only the nodes that will appear on the workspace (after you exit the Workspace Designer and then open the workspace you are now designing).

4. Customize the new workspace as needed. Use one or more of the methods described in the Customizing a workspace section that follows to:
   - Make a node appear on a workspace.
   - Remove a node from a workspace.
   - Set which servers or databases appear on a workspace.
   - Set which database objects appear on a workspace.
5. Select **Workspace**, **Save**.
6. Close the Workspace Designer window.

Customizing a workspace

To customize a workspace, use any one or more of the following methods.

Make a node appear on a workspace

This procedure assumes you have accessed the Workspace Designer (as described in steps 1 and 2 of *Create a new workspace* on page *4-3* or steps 1 through 3 of *Alter a workspace* on page *4-4*).

1. Select a node in the left window and then click the **Enable** push button (see the picture on page *4-3*).

   The switch on the **Enable** push button moves to the "up" position, the node turns from blue to black (if you have a color monitor), and the node appears on the window on the right.

   Enabling (or disabling) a node also enables (or disables) its child nodes.

2. Repeat step 1 as needed.

Note: You can shift-click to select multiple nodes. To easily select or deselect all the nodes on the workspace, you can use the **Select All** and **Clear Selection** push buttons. For example, click **Select All** and then **Enable** to turn on all the nodes on a workspace.

---

3. Save your change and exit the Workspace Designer as described in *Create a new workspace* on page *4-3* or *Alter a workspace* on page *4-4*.

Remove a node from a workspace

This procedure assumes you have accessed the Workspace Designer (as described in steps 1 and 2 of *Create a new workspace* on page *4-3* or steps 1 through 3 of *Alter a workspace* on page *4-4*).

1. Select a node in the left window and then click the **Enable** push button (see the picture on page *4-3*).

   The switch on the **Enable** push button moves to the "down" position, the node turns from blue to black (if you have a color monitor), and the node disappears from the window on the right.

   Disabling (or enabling) a node also disables (or enables) its child nodes. For example, if you disable the Server node, all database nodes and Statistics nodes are also disabled.

Note: You can shift-click to select multiple nodes. To easily select or deselect all the nodes on the workspace, you can use the **Select All** and **Clear Selection** push buttons. For example, click **Select All** and then **Enable** to turn on all the nodes on a workspace.

2. Repeat step 1 as needed.
3. Save your change and exit the Workspace Designer as described in *Create a new workspace* on page *4-3* or *Alter a workspace* on page *4-4*.

Set which servers or databases appear on a workspace

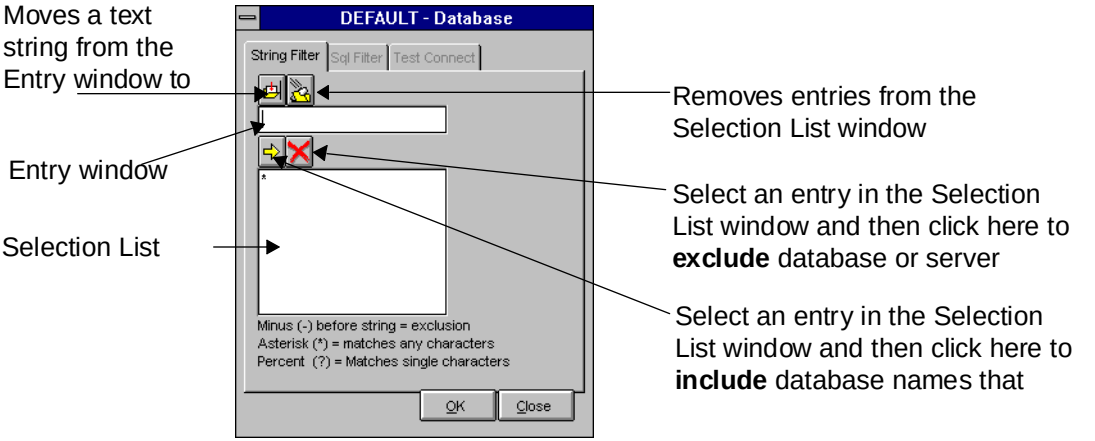
You can explicitly include or exclude particular servers or databases from a workspace. For example, you may want to show databases TEST1 and TEST2 on your workspace, but not TEST3.

This procedure assumes you have accessed the Workspace Designer (as described in steps 1 and 2 of *Create a new workspace* on page *4-3* or steps 1 through 3 of *Alter a workspace* on page *4-4*).

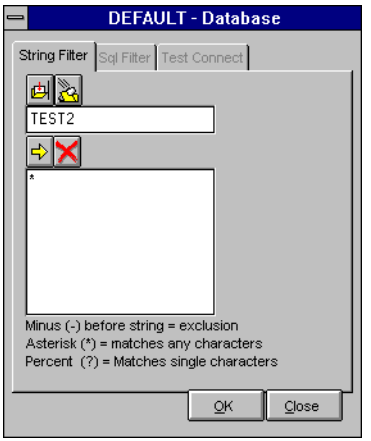
Note: The Connect Manager controls which servers and databases appear by default on all workspaces. In order for a server (and, by default, all the installed databases on it) to appear on a workspace, a connection to the server must exist in the Connect Manager. Once the connection is established, you can use the Workspace Designer to more precisely filter which servers or databases appear on a specific workspace.

1. Right-click a server or database node, then select **Properties**.

   A database or server property sheet appears.

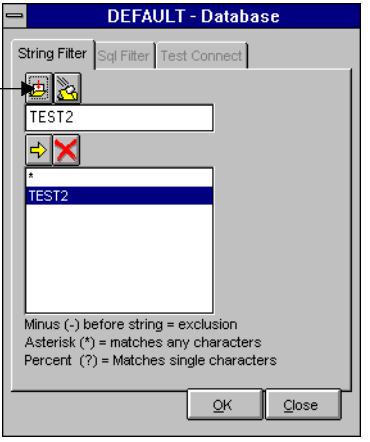


2. Enter a text string (for example, TEST2) in the Entry window.



3. Click the push button that moves the entry down to the Selection List window.

Click here.



4. To exclude databases (or servers, if you are working on the server property sheet) that match the text string you entered, click the push button with an "X." To include databases (or servers) that match the text string, click the push button with an arrow (this is the default).

   For example, click the push button with an "X" to exclude database TEST2 from your workspace.

   As noted on the property sheet, you can use the asterisk (*) and percent (%) characters in the text strings.

5. Repeat steps 2 and 4 as needed.

6. Click **OK** to exit and save your changes, or **Close** to exit and discard your changes.
7. Save your change and exit the Workspace Designer as described in *Create a new workspace* on page *4-3* or *Alter a workspace* on page *4-4*.

Set which database objects appear on a workspace

You can explicitly include or exclude certain database objects from a workspace. Here we show you how to set which tables appear on a workspace, but you can use this procedure to include or exclude any of the following:

- Users
- Tables
- Columns
- Indexes
- Views
- Synonyms
- Stored commands
- Stored procedures
- External Functions
- Events
- Triggers

This procedure assumes you have accessed the Workspace Designer (as described in either steps 1 and 2 of *Create a new workspace* on page *4-3* or steps 1 through 3 of *Alter a workspace* on page *4-4*).

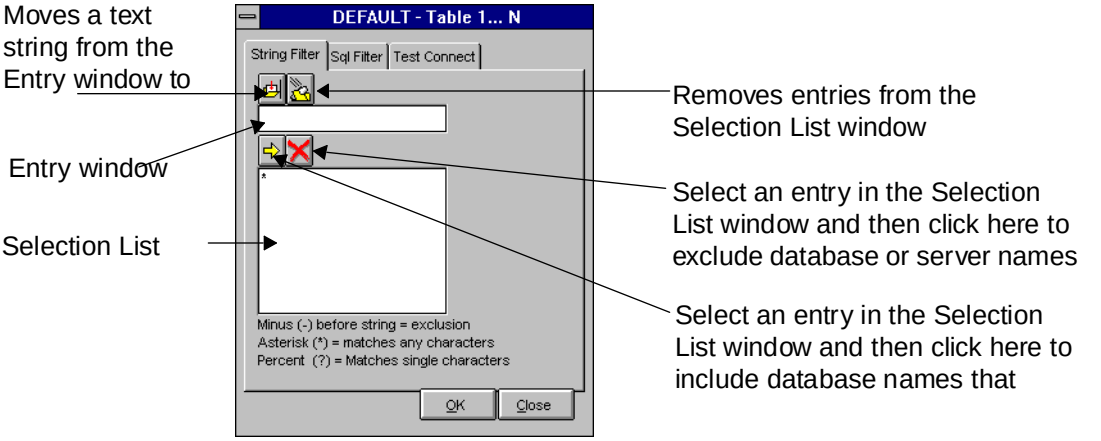
1. Right-click any 1...N database object node. For example, right-click the Tables 1...N node.

Note: Each 1...N node in the Workspace Designer is a generic representation for all the individual instances of that database object that will eventually appear on the workspace, after you open it outside the Workspace Designer. For example, the Tables 1...N node represents all the table nodes.

---

2. Select **Properties**.

A database or server property sheet appears.



3. Enter a text string in the Entry window, then click the push button that moves the entry down to the Selection List window (refer to the picture). For example, type TABLE2.
4. To exclude database objects that matches the text string you entered, click the push button with an "X." To include database objects that matches the text string, click the push button with an arrow (this is the default).

   For example, click the push button with an "X" to exclude table Table2 from the workspace.

   As noted on the property sheet, you can use the asterisk (*) and percent (%) characters in the text strings.
5. Repeat steps 3 and 4 as needed.
6. Click **OK** to exit and save your changes, or **Close** to exit and discard your changes.
7. Save your change and exit the Workspace Designer as described in *Create a new workspace* on page *4-3* or *Alter a workspace* on page *4-4*.

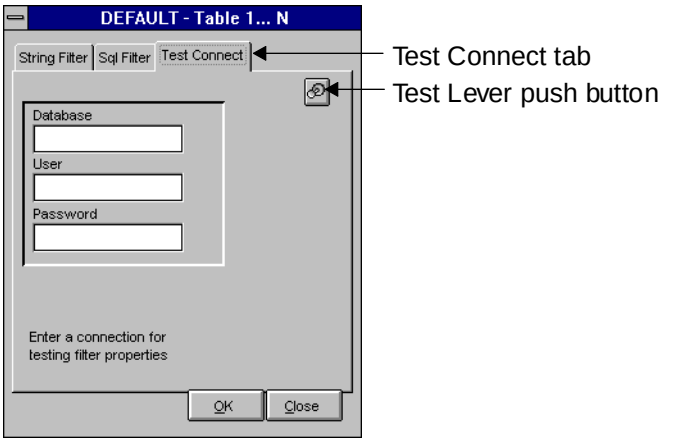
Set which database objects appear on a workspace—Alternative SQL filter method

This method is identical to the preceding example, except that it enhances performance because it edits the WHERE clause that SQLConsole uses to populate the database object nodes on a workspace. The preceding example, on the other hand, applies additional restrictions *after* a SQL query gets information from the database about which objects to place on the workspace.

Use this method only if you are familiar with SQL. If you make an error, your workspace will fail to populate with objects and data.

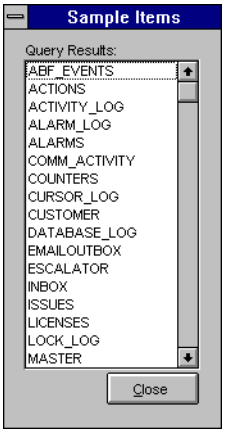
As in the preceding example, we use the Tables 1...N node as an example, but you can use any 1...N node.

1. Perform steps 1 through 2 of the preceding procedure (Set which database objects appear on a workspace).
2. Click the Test Connect tab of the property sheet that appears.

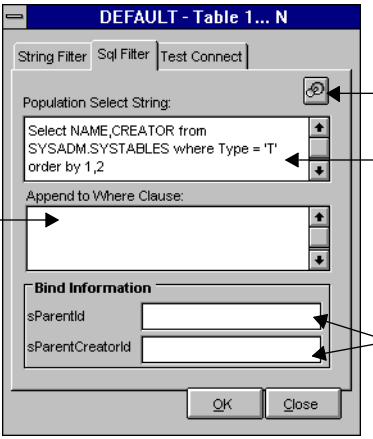


3. In the **Database** field, enter the name of the database.
4. Enter the user name and password.
5. Click the Test Lever push button to test your connection to the database.

   A dialog box indicates your connection is successful.

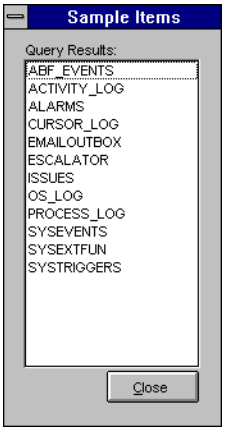


6. Close the Sample Items dialog box.
7. Click the SQL Filter tab.



8. Enter edits to the WHERE clause in the Append to Where Clause window, then click the Test Lever push button to test the query.
9. Save your change and exit the Workspace Designer as described in *Create a new workspace* on page *4-3* or *Alter a workspace* on page *4-4*.

   For example, if you perform this procedure against the SQLCON database, the following results appear after you enter `COLCOUNT>10 AND PAGECOUNT=2` and click the Test Lever push button.



10. Close the Sample Items dialog box.
11. If you want, enter information in the bind variable fields.

12. Repeat steps 8 and 11 as needed until you are satisfied with the results of your query.
13. Click **OK** to exit and save your changes, or **Close** to exit and discard your changes.

Chapter 5

SQLConsole Tools

This chapter provides an overview of the SQLConsole tools:

- Calendar.
- Connect Manager.
- SQLTalk.
- Log Manager.
- Event Manager.
- Scheduling Manager.
- Mail Manager.
- Stored Procedure Manager.
- SQLConsole Settings.

Viewing the SQLConsole tools

To view a SQLConsole tool, double-click its node on a workspace. For example, to view the Calendar, double-click the Tools node to open it, then double-click the Calendar node.

To see how to open a workspace so you can view the tools, read *Opening a workspace* on page *3-4*.

Note: If you are in standard mode, only a subset of tools appears on your workspace or workspaces.

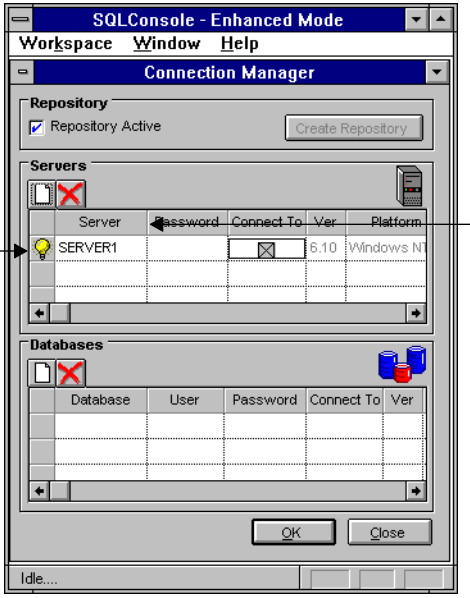
Calendar

You can use the Calendar to monitor either or both of the following:

- Database maintenance events that you schedule with the Scheduling Manager (see the Calendar information in *Chapter 10, Automating Maintenance with the Scheduling Manager*).
- Server conditions that you track with the Alarm Manager (see the Calendar information in *Chapter 9, Monitoring Server Conditions with the Alarm Manager*).

Connect Manager

The Connect Manager determines the servers and databases to which you connect. Your Connect Manager settings apply to *all* workspaces, and any change you make to them takes effect when you restart SQLConsole.



A lit light bulb indicates an active connection.

You can place the cursor at the edge of a column heading and drag to resize it.

What appears in the Connect Manager?

If you have a server connection, you also have a possible connection to all the available databases on that server. By default, these databases appear on your workspaces but do *not* appear in the Connect Manager. Instead, a database appears in the Connect Manager (in the Databases group box in the bottom half of the Connect Manager window) only if you explicitly enter a row with the database information.

You can connect to a database even if you do not have a connection to the server on which it is installed (read *Connecting a database* on page *6-7*).

Disabling connections

You can use the Connect Manager not only to establish connections, but also to prevent SQLConsole from connecting to a server or database. For example, if a particular database will be offline for a several weeks, you can tell SQLConsole to bypass making a connection to it (read *Connecting a database* on page *6-7*).

Interaction with the Workspace Designer

You can use the Workspace Designer to prevent a server or database from appearing on a workspace, even if you have a connection to it. For example, if you have a connection to a server (SERVER1, for example), you may also have a connection to each installed database on it (DB1, DB2, and DB3). You can, however, use the Workspace Designer to filter which databases appear on your workspace, so only DB1 and DB2 appear on your workspace.

For more information...

Read *Chapter 6, DBA Operations*, for more detailed information on connecting and disconnecting to servers and databases.

SQLTalk

SQLTalk is an interactive user interface for managing SQLBase databases. SQLTalk has a complete implementation of SQL and many extensions.

If you are familiar with SQL or have extensive SQL scripts that you use with SQLBase, you may prefer to use SQLTalk for certain database maintenance tasks. You can use SQLTalk to:

- Execute SQL commands.
- Connect or disconnect cursors and databases.
- Perform database administration functions such as BACKUP, RESTORE, LOAD, and REORGANIZE.
- Store, run, and erase SQL stored commands and procedures.
- Run scripts of SQLTalk commands.
- Format, display, and print the results of a SQL query in a complex multi-page report.

For more information, refer to online help for SQLTalk or the *SQLTalk Command Reference*.

Note: Any isolation level you set while running SQLTalk from SQLConsole applies to your entire process (SQLConsole). Remember this, so you do not cause SQLConsole to leave unintended locks on databases.

Log Manager

With SQLConsole, you can collect statistics on a variety of server and database attributes. To use the statistics information most effectively, log the information to

files and then examine the results over time. This allows you to use your own system's performance as a baseline and to objectively measure the results of any tuning changes you make.

You can monitor SQLBase statistics without using the Log Manager (by opening the nodes under the Statistics node on a server or database), but using the Log Manager is recommended because it lets you capture your system's performance over a period of time under its usual operating conditions.

For more information on the Log Manager, read *Logging information statistics* on page *7-13*.

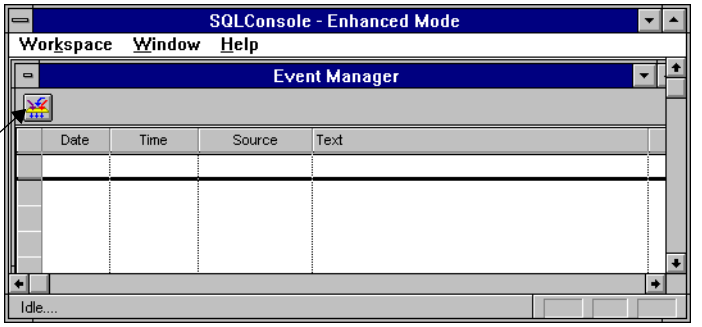
Scheduling Manager

With the Scheduling Manager, you can make SQLBase automatically execute database backups, scripts, and other database maintenance tasks. For more information, read *Chapter 10, Automating Maintenance with the Scheduling Manager*.

Event Manager

Whenever the Scheduling Manager or the Alarm Manager logs information, the Event Manager window opens.

Use the Filter push button to set the information that appears in the Event Manager.



You can control the amount of information in the Event Manager by clicking the **Filter** push button and entering information in the Event Filter dialog box.

Mail Manager

You can use the Mail Manager to send mail to a user from SQLConsole. This functionality makes it easy to notify users of server events, such as an impending shutdown.

You can send mail when the MAIL_ID cursor parameter of the recipient cursor is set. Although a user's application can set this through a *sqlset* call, you can set a user's MAIL_ID globally in the user's WIN.INI file by creating a SQLBASE heading. If the SQLBASE heading already exists, add the MAIL_ID section to it.

For example, in the following portion of a WIN.INI file, "Corina Lazo" is the user's e-mail name:

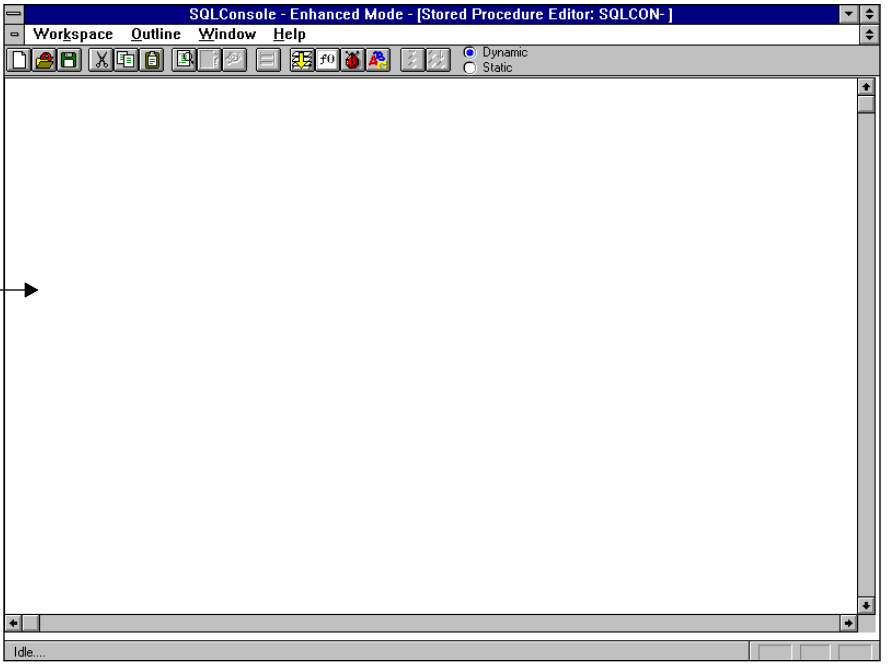
```
[SQLBASE]
MAIL_ID=Corina Lazo
```

The Windows SQL/API library reads the WIN.INI file, sees the setting, and sets the cursor parameter for all connected cursors.

Stored Procedure Manager

Stored procedures can improve performance and simplify your applications by moving flow control language and SQL statements to the server.The Stored Procedure Manager is an editor that lets you create or alter stored procedures within SQLConsole.

To open the Stored Procedure Manager, double-click its node (under the Tools node) on a SQLConsole workspace.



Enter stored procedure code in this window.

Note: For general information on stored procedures, refer to the *SQLBase SQL Language Reference*.

Tool bar

Use the push buttons on the tool bar to help you edit a stored procedure. To identify a push button, put the cursor over it and wait a few seconds for the push button's label to appear. At the same time, the status bar in the bottom left of the Stored Procedure Manager window shows the push button's description.

Outline window

Write code in the Outline window. You name a procedure, specify the data type and name for procedure parameters and local variables, and enter the actions for the procedure.

You must follow certain convention for the indentation and arrangement of stored procedure code. For more information, refer to the *SQLBase SQL Language Reference*.

Execution table

When you store a stored procedure with parameters, an execution table appears that lets you enter values for the function parameters and look at the results that the stored procedure returns. This is a useful way to test stored procedure logic.

Each parameter in a stored procedure appears as the name of a column in the table. Put a value in the input row underneath the column header and press the **Execute** push button (the push button with a test lever handle on it). This returns the results in the table below the input line. If a stored procedure is written to return a result set, all the rows of the result set appear in the table.

SQLConsole Settings

Depending on your requirements, you may want to change certain default SQLConsole options. You do this with the SQLConsole Settings node on a SQLConsole workspace, which is visible under the Tools node.

When you change a setting, the new setting takes effect immediately. The new setting is saved when you close the SQLConsole session, so subsequent sessions also use the new setting.

In general, you may want to change one or more SQLConsole settings if you want to:

- Change the rate at which the SQLConsole activity screens refresh.
- Switch between standard and enhanced mode.
- Log information about server activity (see *Logging information statistics* on page *7-13*).
- Use SQLConsole with your e-mail system.
- Configure SQLConsole to work with a pager.

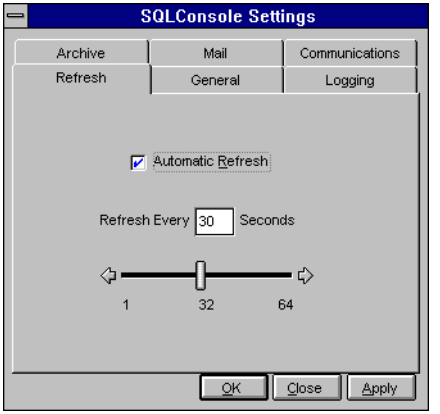
Note: Any change you make to a SQLConsole setting applies to *all* workspaces.

Change a SQLConsole setting

1. Double-click the SQLConsole Settings node (under the Tools node on a workspace).

Note: If you need to open the Tools node, double-click it. Double-clicking alternately opens and closes a node.

The Settings property sheet appears.



2. Click the appropriate tab (**Logging**, for example).
3. Change a setting or settings, then save your changes and close the window by using the push buttons:
   - **OK** — Applies change and closes the property sheet.
   - **Close** — Closes the property sheet.
   - **Apply** — Applies change but does not close property sheet.

Description of settings

Use the guidelines in the following table to change settings.

Tab	Description
Refresh	• Automatic Refresh — If a check appears in the check box, automatic refresh is on (enabled). If no check appears it is off (disabled). Note that the **Automatic Refresh** push button (the push button that looks like a clock) turns automatic refresh on or off in a given window. • Refresh scroll bar — If automatic refresh is enabled, use this scroll bar to control the rate at which SQLConsole refreshes all SQLConsole windows. To change the rate, drag the scroll bar to the desired location, from 1 to 64 seconds. Alarms are checked at this refresh rate, and are checked even if automatic refresh is disabled. You can selectively turn automatic refresh off in individual information windows while letting it continue to run in others.

Tab	Description
General	Use this to set: • Standard mode. Standard mode (unlike enhanced mode) disables certain automated SQLConsole functionality in order to speed performance at startup. • Whether to confirm overwrite of stored procedures. • Visible modem control—If set when you use SQLConsole with a pager, displays the progress of a call. Note that SQLConsole does not perform error reporting of modem or phone or pager host problems.
Logging	Use this to capture server statistics in log files.
Archive	Use this to specify how and where to keep archives of the server activity logs (if you log server statistics; see the **Logging** tab).
Mail	Configures SQLConsole to work with your mail system. Use this option if you want to use e-mail to notify a user if a particular action occurs, or if you want SQLConsole's Alarm Manager to automatically send e-mail if a particular action occurs. • Mail — Your mail system. SQLConsole is supported only with Microsoft Mail. • Logon Name — The mail user. You may want to set up a separate user for SQLConsole's use. • Password — The password for the user. • P.O. Path — The path of your post office.

Tab	Description
Communications	Configures SQLConsole to work with a pager. Use this option if you want SQLConsole's Alarm Manager to notify a user or users by pager if a particular action occurs. Modem Type — Only Hayes command set compatible modems are supported. Baud Rate — From 300 to 38400 baud terminal. Com Port — COM1 to COM4. Timeout — Time in seconds. Pager Pause — Contains modem commands to pause the local modem until a connection is complete. Pager Terminate — Number to dial to the pager company to complete the page. Usually this is the pound (#) key. Line (Tone or Pulse) — Set to the line type supported by the phone service to you. (Note also the Visible Modem Control check box on the General tab.) Note that SQLConsole does not perform error reporting of modem or phone or pager host problems and thus does not retry failed connections.

Chapter 6

DBA Operations

This chapter describes how to perform common DBA (database administration) tasks with SQLConsole:

- Connecting a server.
- Disconnecting a server.
- Shutting down a server.
- Enabling a server.
- Terminating a server.
- Creating a database.
- Connecting a database.
- Deleting a database.
- Installing a database.
- Deinstalling a database.
- Shutting down a database.
- Enabling a database.
- Backing up a database.
- Restoring a database.
- Partitioning a database.

Connecting a server

When the DEFAULT workspace first opens it only displays the Tools node. For security reasons, SQLConsole makes no attempt to automatically locate or connect to servers or databases. Every server which SQLConsole will be used to manage must be added to the connect tree by means of the Connection Manager (under the Tools node). You may also use the Connection Manager to tell SQLConsole the user name and password to be used to connect to databases on those servers.

You can also use the Connection Manager to tell SQLConsole to bypass making a connection to a server or database (see the sections Disconnecting a database or Disconnecting a server in this chapter).

Note: Your Connection Manager settings apply to *all* workspaces.

Connecting to a server enables the following operations:

- Backing up, recovering, and restoring a database and its logs.
- Creating and deleting a database and its logs.
- Installing and deinstalling a database.
- Monitoring the server.
- Alarm notification.

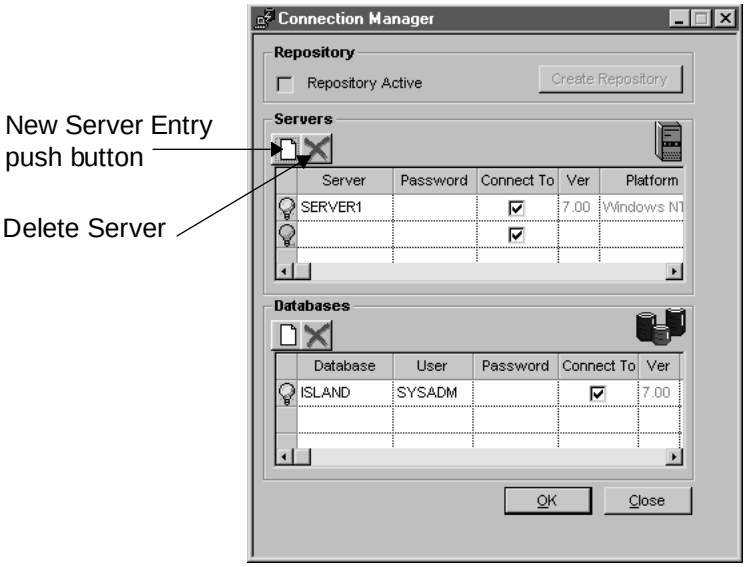
A server connection is *not* required in order to use the Scheduling Manager.

Note: If you use the Connection Manager to connect to a database for which you do not have a server connection, the Connection Manager displays the associated server information for that database. This row of server information, though, is for your information and is not an actual server connection.

Connect to a SQLBase server

1. Open a workspace (for more information, read *Opening a workspace* on page *3-4*).
2. Double-click the Connection Manager node (under the Tools node).

The Connection Manager window opens. The Server window shows a list of all connected servers.



3. Click the **New Server Entry** push button.

   An outlined row appears for you to insert information.

4. Type the server name.

5. Enter a password in the Password field.

   If the server has a password defined in the SQL.INI file, use that value. If you do *not* have a server password defined in SQL.INI, then any value in the password field will do, including a null password. However, we recommended that you set a password for a SQLBase server. For example:

   ```
   [DBWSERVR]
   SERVERNAME=SERVER1
   PASSWORD=2MINNEOLA
   ```

6. If you do not want SQLConsole to connect to the server, click to remove the check from the Connected To check box. Otherwise, by default SQLConsole is enabled to connect to the server.

   SQLConsole automatically enters the information in the remaining fields.

7. Click **OK**.

8. Exit from and then restart SQLConsole to see the new server connection.

Note that SQLConsole is connected to a server only if it appears in the Servers group box *and* the Connected To check box is checked.

Note: If you are connected to a server, you will also see the installed databases on that server, even though a specific row for each database does not appear in the database section of the Connection Manager.

Disconnecting a server

Disconnecting a server removes it from the list of connected servers. All monitoring and alarm notification is halted and any windows open for the server are closed.

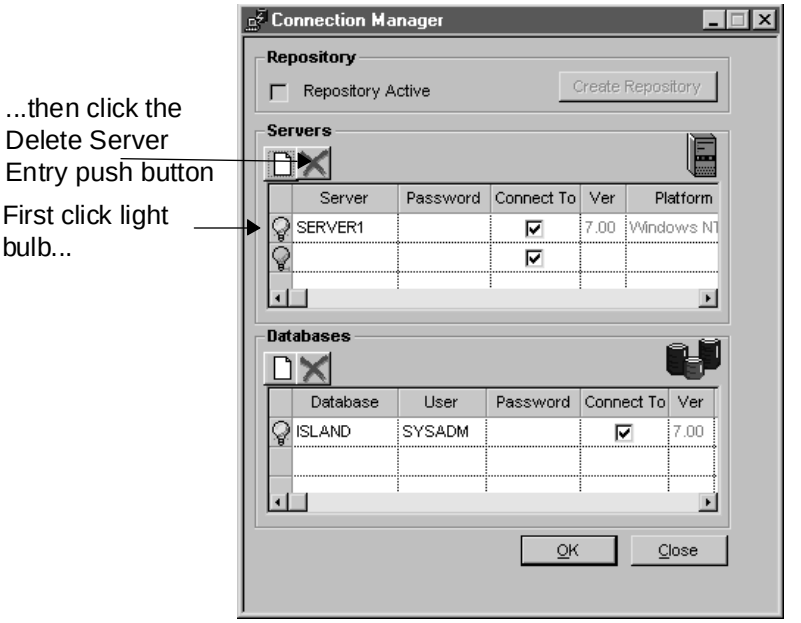
You must restart SQLConsole to see the effect of disconnecting from the server. After you disconnect a server and restart SQLConsole, the databases on the server do *not* appear on a workspace unless you explicitly connect to them in the Databases group box (as described in *Connect to a database* on page *6-8*).

Disconnect a server

1. Open a workspace (for more information, read *Opening a workspace* on page *3-4*).
2. Double-click the Connection Manager node (under the Tools node on a SQLConsole workspace).

   The Connection Manager window opens.

3. Click the light bulb icon that appears on the left of the server information row for the server you want to disconnect, then click the **Delete Server Entry** push button.



4. Exit from and then restart SQLConsole.

Shutting down a server

Shutting down a server prevents new connections to it. No one can connect to the server after you shut it down (except for a server handle), but users who are already connected can continue their work.

To bring a server back online after a shutdown, enable the server.

Shut down a server

1. Open a workspace (for more information, read *Opening a workspace* on page *3-4*).
2. Right-click a server node in a SQLConsole workspace.
3. Select **Shutdown** from the context menu.
4. In the Shutdown Server dialog box, select **Shutdown**.

Enabling a server

This command allows a server that has been shut down to accept new connections.

Enable a server

1. Open a workspace (for more information, read *Opening a workspace* on page *3-4*).
2. Right-click a server in a SQLConsole workspace.
3. Select **Enable** from the context menu.

Terminating a server

Terminating the server is equivalent to pressing the **Escape** key on the server machine's keyboard. Connected user sessions are terminated and open transactions are left uncommitted. When the server is subsequently started and a database connection occurs, crash recovery occurs and open transactions roll back.

Note that after the server is terminated, the databases on it are no longer available.

Terminate a server

1. Open a workspace (for more information, read *Opening a workspace* on page *3-4*).
2. Right-click a server in a SQLConsole workspace.
3. Select **Terminate** from the context menu.

Creating a database

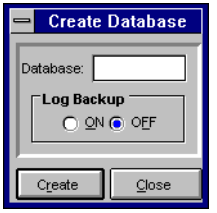
This operation physically creates, installs, and brings online a database. SQLBase creates the database in a subdirectory of the directory specified by the DBDIR keyword in the server's SQL.INI file. The default for DBDIR is \SQLBASE.

This operation requires a server connection.

Create a database

1. Open a workspace (for more information, read *Opening a workspace* on page *3-4*).
2. Right-click a server in a SQLConsole workspace.
3. Select **Manage Database**, **Create** from the context menu.

4. In the Create Database dialog box, enter the new database name.



   The maximum length of a database name is eight characters. Unlike ordinary identifiers, you cannot use special characters (such as $, #, @, and __) in a database name, and the first letter must be alphabetic.

   Do not specify an extension for a database name, such as DEMO.XYZ. SQLBase automatically assigns a database name extension of .DBS. Do not specify the name MAIN for a database, since this is used to store control information for partitioned databases. Also, do not specify the name of an existing server for a database.

5. If you want log backup enabled, click **On** in the Log Backup group box.

6. Click **Create**.

   It may take 30 or more seconds to complete this operation. When the database is created, it appears on your workspace. To enable the database so operations can be performed on it, you have to tell SQLConsole the username and password to use when connecting to the database. To do this, either right-click on the database name and choose security from the pop-up menu, or go to the Connection Manager and add a new line for the newly-created database. The change takes effect after SQLConsole is restarted.

Connecting a database

If you have a server connection (for example, a connection to a local SQLBase server when you start SQLConsole), you also have a connection to (and can thus perform actions on) all the available databases on that server. These databases appear on your workspace or workspaces, but they do *not* appear in the Connection Manager.

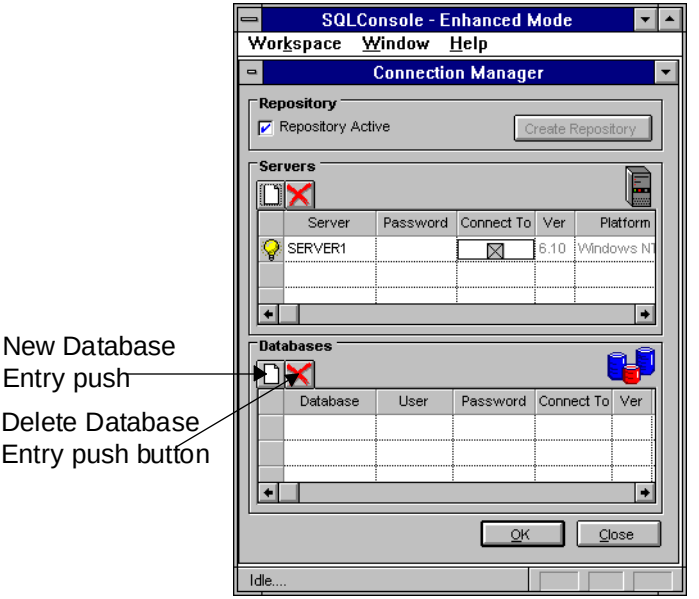
Use the Connection Manager to connect to a database only if you do not have a server connection in the Connection Manager. For example, if you want to connect to the SALES database but not the server on which SALES is installed, connect to SALES in the Databases group box.

When you connect to a database in the Connection Manager, the change takes effect after you restart SQLConsole.

Connect to a database

1. Open a workspace (for more information, read *Opening a workspace* on page *3-4*).
2. Double-click the Connection Manager node (under the Tools node).

   The Connection Manager window opens.



3. Click the **New Database Entry** push button.

   An outlined row appears for you to insert information.
4. Type the database name.
5. Enter the user name in the User field. The default is SYSADM.
6. Enter the password in the Password field.
7. If you do not want SQLConsole to connect to the database, click to remove the check from the Connected To check box. Otherwise, by default SQLConsole will connect to the database.
8. Click **OK**.
9. Restart SQLConsole to see the effect of your change.

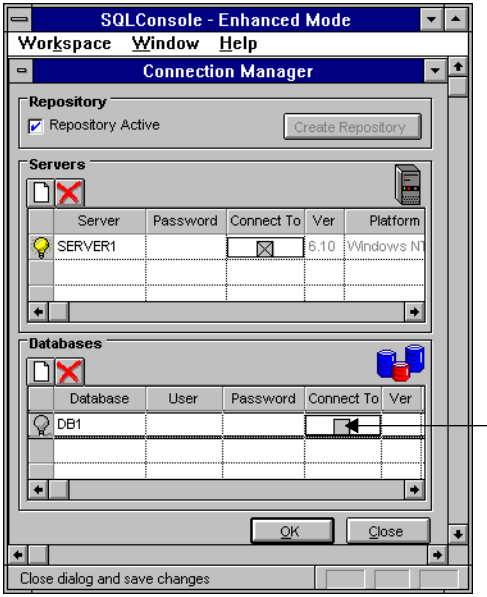
Disconnecting a database

Use the following procedure to prevent SQLConsole from connecting to a database. For example, if a database will be offline for several days, you can prevent SQLConsole from trying to connect to it.

Disconnect a database

1. Open a workspace (for more information, read *Opening a workspace* on page *3-4*).
2. Double-click the Connection Manager node (under the Tools node).

   The Connection Manager window opens.
3. Enter a row for the database in the Databases group box (if a row for the database does not already exist). Use the procedure in step 3 through step 6 in *Connect to a database* on page *6-8*.
4. Next, click to remove the check from the Connected To check box.



When you restart SQLConsole, the database appears in grey on all workspaces.

Deleting a database

The delete database operation physically deletes the entire database directory of a database. The database and all associated log files are removed. If log files have been redirected to a different drive, the log directory also is removed.

This operation requires a server connection.

Delete a database

1. Open a workspace (for more information, read *Opening a workspace* on page *3-4*).
2. Right-click the database you want to delete.
3. Select **Manage**, **Delete** from the context menu.
4. In the Delete Database dialog box, select **Delete**.

   SQLConsole checks whether users are connected to the database you want to delete; if so, you receive an error. If you are successful, the database name disappears from all workspaces.

Installing a database

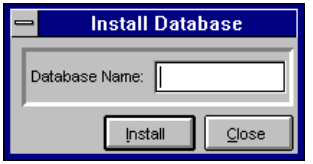
This operation installs a database on the current server and adds a DBNAME keyword for it in the server's SQL.INI file. Before you can install a database, it must exist on the server's disk in an appropriate subdirectory.

This command requires a server connection.

Install a database

1. Open a workspace (for more information, read *Opening a workspace* on page *3-4*).
2. Right-click the node for the server that contains the database.
3. Select **Manage Database**, **Install** from the context menu.

   The Install Database dialog box appears.



4. Enter the name of the database to install, then click **Install**.

Deinstalling a database

This operation removes a database from the server and removes its corresponding DBNAME entry from the SQL.INI file. This operation does not physically delete the database. You can deinstall a database only if there are no connections to it.

This operation requires a server connection.

Deinstall a database

1. Open a workspace (for more information, read *Opening a workspace* on page *3-4*).
2. Right-click the database you want to deinstall.
3. Select **Manage**, **Deinstall** from the context menu.
4. In the Deinstall Database dialog box, select **Deinstall**.

   To re-install the database on the server, install the database.

Shutting down a database

Shutting down a database prevents new database connections. No one can connect to the database after it is shut down, except for a DBA, but users already connected can continue their work.

Only SYSADM can shut down a database.

After this function completes, anyone trying to connect to the database receives a "shutdown in progress" message. All current users remain connected and all current transactions continue.

To bring a database back online after it has been shut down, you need either to deinstall and install the database, or to enable it.

Shut down a database

1. Open a workspace (for more information, read *Opening a workspace* on page *3-4*).
2. Right-click the database you want to shut down.
3. Select **Shutdown** from the context menu.
4. In the Shutdown Database dialog box, select **Shutdown**.

Enabling a database

Enabling a database that has been shut down lets users establish new connections to the database.

Enable a database

1. Open a workspace (for more information, read *Opening a workspace* on page *3-4*).
2. Right-click the database you want to enable.
3. Select **Enable** from the context menu.
4. In the Enable Database dialog box, click **Enable**.

Loading and unloading a database

A load operation can either restore data from a file created with a SQLBase unload operation, or a load operation can enter data into the database from an external file.

Unloading data involves moving database information to a flat file. You can then load this information to a new database.

SQLBase uses the server to perform load and unload processes. This relieves the client of unnecessary processing, reduces network traffic, and allows the server to load and unload the data locally on its own machine.

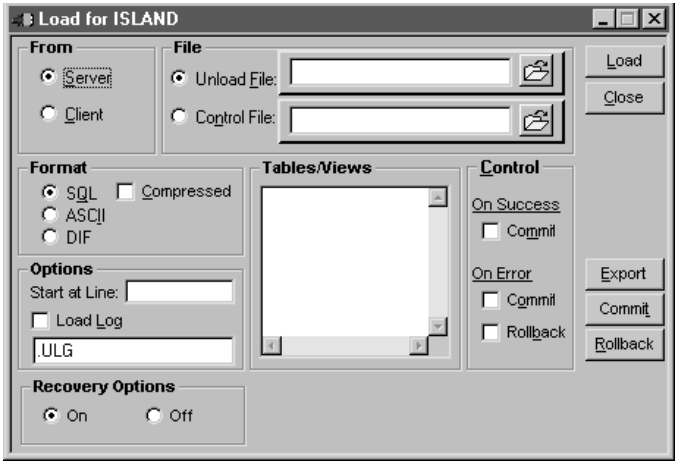
SQLBase load and unload operations handle large amounts of data, especially for large tables or databases. Load and unload files consume storage space and can be expensive to transmit. There is a compression option to reduce the size of these external files. You can compress the external file when you unload it, and decompress it when you reload it. Compression occurs at the server before transmission over the network.

Note: If you perform a load operation with SQLTalk, you can issue a SET RECOVERY OFF command to enhance load performance (as discussed in the *SQLBase Database Administrator's Guide*). You are unable to issue this command from the SQLConsole load dialog box.

Load a database

1. Open a workspace (for more information, read *Opening a workspace* on page *3-4*).
2. Right-click the database into which you want to load data.
3. Select **Manage**, **Load** from the context menu.

The Load dialog box appears.



4. Enter the appropriate information in the load dialog box. For more information, refer to the table that follows this procedure.
5. Click **Load**.

   During the load operation, autocommit is temporarily disabled by SQLBase.
6. When the load operation terminates, use the **Commit** or **Rollback** push buttons to manually commit or rollback the session, or use the settings in the Control group box to automatically commit or rollback.
7. Click **Close** to exit the dialog box.

Selecting load options

The following table describes the options in the load dialog box.

Load Option	Description
From	Use to indicate whether the files are located on the client (local user) or database server machine.

Load Option	Description
File	Use to indicate whether to load data from a single file or from multiple file segments. • Unload File — Loads data from a single file whose name you enter in the data field to the right (you can use the file browse button to locate the file). • Control File — Use with a load control file name if you are loading information from multiple file segments where the unload operation was performed using a control file. Enter the name of the load control file or use the **Browse** push button to locate the file.
Format	Select the format of the file to be loaded: • SQL — Specifies that the load file is a series of SQL commands and was probably created by an UNLOAD command. If the file is compressed, check **Compressed**. A SQL format file contains the CREATE TABLE and CREATE INDEX data definition commands along with corresponding INSERT commands for each table. Since the load file contains the CREATE TABLE command, SQLBase creates the table or tables and associated indexes, so the tables *cannot* yet exist. The command LOAD SQL is equivalent to running the file as a script. The data rows are inserted using bind variables • ASCII — Specifies a load file that contains input data organized in ASCII format. You can load to only one target table. ASCII format is similar to the data format in SQL except that the character fields are delimited by double quotes ("). You cannot load LONG data type columns in ASCII. • DIF — Specifies a load file that contains input data organized in Data Interchange Format (DIF), which is a common format for spreadsheets and databases. Only one table can be loaded from a single DIF file.
Options	• Start at Line — Use to start the load operation from a specific line in the load input file. Select **Load Log** in conjunction with this option, since the log file displays the line number where the error occurred. • Load Log — Use to automatically create a log file. The log file records activities and errors that occur during the load operation. Errors are logged with the line number at which the error occurred. After you fix an error, use the line number with the START AT LINE option to restart the load operation at that point in the source file.

Load Option	Description
Tables/Views	When loading an ASCII or DIF file, use this list box to select the target table or view to which the data will be loaded.
Control	Use to specify the action to occur if the load operation is successful or has an error: • On Success — Select **Commit** to automatically commit if successful. During the load, autocommit is disabled, so failing to commit could lose the results of the operation. • On Error — Indicate whether you want to commit or rollback if an error occurs. Use **Commit** if you are generating a log file, so you can pick up the transaction at the point of error using the START AT LINE option.

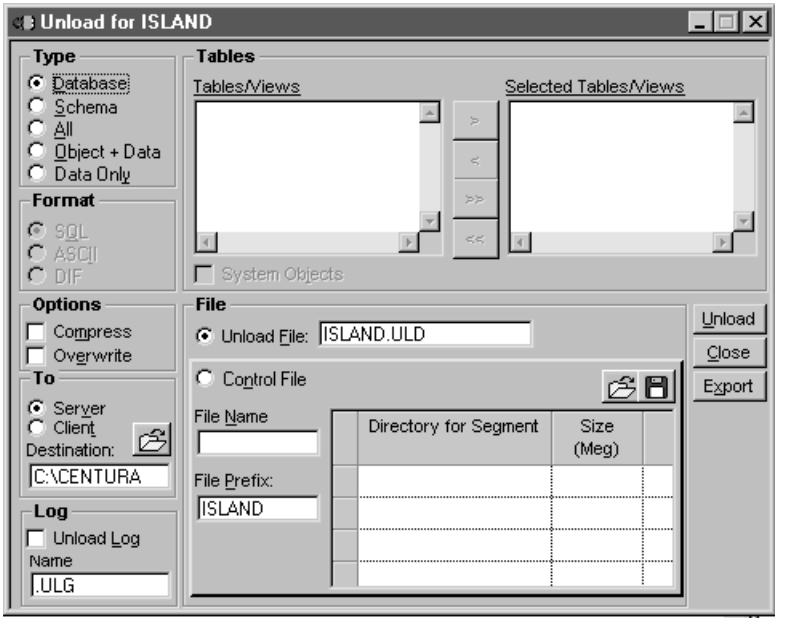
Unload a database

An unload operation allows you to back up a database or transfer data from a database to another program through interchange formats.

An unload operation does not unload invalid stored commands.

1. Open a workspace (for more information, read *Opening a workspace* on page *3-4*).
2. Right-click the database you want to unload.
3. In the pop-up properties box, select **Manage**, **Unload**.

The Unload dialog box appears.



4. Enter the appropriate information in the dialog box. For more information, refer to the table the follows this procedure.

5. Click **Unload**.

Note: You can use the **Export** push button to create the unload command that SQLBase would use and save the command to a file (without actually unloading the database).

---

6. Click **Close** to exit the dialog box.

Selecting unload options

The following table describes the options in the unload dialog box.

Unload Option	Description
Type	Use this setting to indicate what data will be exported. The choices are: • Database — The default. Unloads the entire database to which a user is connected. You must be logged on as SYSADM to give this command. • Schema — Similar to the Database option (above), but unloads only DDL commands (unloads the schema without unloading the data). • All — Unloads all tables and indexes belonging to connected user (you can explicitly connect as a certain user in the Connection Manager). This command does not unload any other database objects. • Object + Data — Unloads the CREATE TABLE statement and the INSERT statements for each row for the object you select. Whether the file format is SQL, ASCII, or DIF can have varying implications for the contents of the file. When you select this option, the Tables/Views window populates to show the database objects. • Data Only — Indicates that only data is written to the external file and no CREATE TABLE or CREATE INDEX statements are written. Whether the file format is SQL, ASCII, or DIF can have varying implications for the contents of the file.
Format	You can use these settings only if you select **Object + Data** or **Data Only** in the Type group box: • SQL — The external file is created with a series of SQL commands. The file contains INSERT commands followed by data rows. • ASCII — The external file contains only data, organized in ASCII format. You can specify only one source table. ASCII format is similar to the data format in SQL except that the character fields are delimited by double quotes ("). You cannot unload LONG data type columns in ASCII. • DIF — The external file contains data organized in Data Interchange Format (DIF), which is a common format for spreadsheets and databases. Only one table can be unloaded in a single DIF file. Only the data in the table is written to the file. You cannot unload LONG data type columns in DIF.

Unload Option	Description
Options	These settings affect the way the output file is written: • Compress — Compress the data when you unload it. This option is not valid for DIF or ASCII data files. • Overwrite — Allows you to overwrite an existing unload file.
To	Indicates whether the files will be written to the client (local user) or the database server. The default is server. You can enter the volume and path in the Destination field, or you can browse the directories of the client or server, whichever is selected, by clicking on the **File Folder** push button.
Log	Creates a log of the unload operation and any errors that occur. You must enter a name for the log file. The log file is created in the same location where the unload file is placed.
Tables	These settings apply only to an Object + Data or Data Only unload. Use this to set which items to include in the unload operation. Items in the left list box represent the tables and views in the database, while items in the right list box represent the objects to be unloaded. Check the System Objects checkbox under the left list box if you want to include system objects in the Tables/Views list box.
File	Depending on the size of the data you need to unload, use one of these two options for unloading the data: • Unload File — Use to unload all data to a single file, whose name you enter. • Control File — Use along with an unload control file name if you are unloading information into multiple file segments. This allows you to unload the data to multiple volumes where the database is too large to fit in a single volume. You must create an unload control file for SQLBase to use by entering a control file name, the prefix of the file segment names, the directory into which to place the segments, and the maximum size of an unload segment file in megabytes. Name as many segments as are required for the size of the unload operation. When you finish entering the information for the control file, save the file using the **Save Control File** push button.

Backing up a database

There are two ways of backing up a SQLBase database: online or offline. You make an offline backup with an operating system utility or copy command after you deinstall the database. You make an online backup with the SQLTalk BACKUP command, a call to a SQL/API function, or by right-clicking a database node and then selecting **Backup** from the context menu that appears.

This section discusses how to perform an online backup using the **Backup** menu item from a database context menu.

For information on backing up a database with the BACKUP command in SQLTalk (which you can do through the SQLTalk node on your workspace), refer to the *SQLTalk Command Reference*.

For information on automating database backup through SQLConsole, read *Scheduling backups* on page *10-6*.

For a general discussion of backup and recovery, refer to the *SQLBase Database Administrator's Guide*.

Backup options

You have three online backup options:

- The BACKUP SNAPSHOT command.

  Backs up only the database file and those log files needed to restore the database to a consistent state. This includes the current active log file, since BACKUP SNAPSHOT forces a log rollover.

  This command is the only BACKUP command that does not require LOGBACKUP to be on. *If* LOGBACKUP is on, back up the log files left in the database directory with a BACKUP LOGS command. SQLBase will then delete them automatically.

Note: You can set the LOGBACKUP option with the SQLTalk SET LOGBACKUP command, the *sqlset* API function, or by right-clicking a database node and then selecting Properties. For more information on using SQLConsole to set it, read *Appendix A, Server and Database Properties*.

- The BACKUP DATABASE command.

  Backs up the database file. You should *never* back up a database without also backing up the log files with it.

- The BACKUP LOGS command.

  Backs up the log files and then deletes them.

BACKUP SNAPSHOT is the recommended way to backup your database and log files because it is easy and provides you with a backup from which you can recover the database in one step.

BACKUP DATABASE and BACKUP LOGS are provided for sites with large databases that want to do incremental backups.

For incremental backups, back up log files using the BACKUP LOGS command. For example, you could back up the database and logs every Sunday, while on Monday through Saturday you could back up only the logs.

A backup directory can be on a client or server computer. Once you have backed up a database and its log files to a directory, you can copy the backup files to archival media and delete the backup files from the client or server disk.

Online backups briefly lock the database during two periods: when the backup begins and the main control page of the database is updated, and when the backup is almost completed and another update takes place.

For more information...

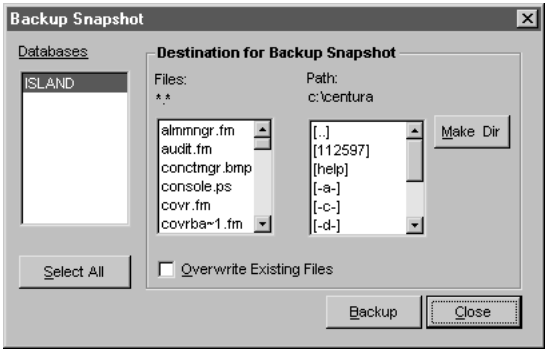
For additional information on backups and backup strategy, refer to *Chapter 10, Automating Maintenance with the Scheduling Manager* and the *SQLBase Database Administrator's Guide*.

Back up a database

If media recovery is important to your site, set the LOGBACKUP parameter on as needed. LOGBACKUP specifies that logs are to be backed up by the DBA before SQLBase deletes them. LOGBACKUP must be on to do a BACKUP DATABASE or BACKUP LOGS operation, but does not need to be on to do a BACKUP SNAPSHOT operation. For more information, refer to the *SQLBase Database Administrator's Guide*.

1. Open a workspace (for more information, read *Opening a workspace* on page *3-4*).
2. Right-click the node of the database you want to back up.
3. Select **Backup** and then any of the following: **Snapshot**, **Database**, or **Logs**.

For example, if you select **Backup**, **Snapshot** the following dialog box appears.



4. Select the database or databases you want to back up.
5. Select the location for the backup files.
6. Click **Backup** to perform the backup, then click **Close** to close the dialog box.

Restoring a database

Restoring a database is the process of recovering a database that has been lost. Data recovery depends on the DBA's backup plan and the frequency with which backups were made.

If a database becomes damaged, you can restore a database backup with SQLTalk's RESTORE command, a call to a SQL/API function, or through the **Restore** context menu on a SQLConsole workspace. This section discusses how to restore a database using a SQLConsole workspace.

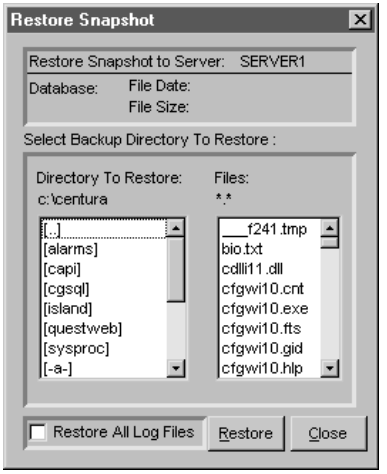
For information on the SQLTalk RESTORE command, refer to the *SQLTalk Command Reference*.

For a general discussion on database recovery and the various recovery options, refer to the *SQLBase Database Administrator's Guide*.

Restore a database

1. Open a workspace (for more information, read *Opening a workspace* on page *3-4*).
2. Right-click the node of the database you want to recover.
3. Select **Restore** and then any of the following: **Snapshot**, **Database**, or **Logs**.

For example, if you select **Restore, Snapshot** the following dialog box appears.



4. Enter information in the dialog box and then click **Restore**.
5. Click **Close** to close the dialog box.

Partitioning a database

The Partition Manager lets you quickly set up partitioned databases on a server. Partitioning a database lets you store a database and its log files across several physical disk partitions. This can have a positive effect on performance since reads and writes to a database can occur in parallel on separate drives. Also, partitioning may be necessary when a database is larger than the largest available disk partition.

To open the Partition Manager, double-click the Partitions node under a server node.

The Partition Manager has a tool bar that contains the commands you need to create and maintain a partitioned database. As you place the cursor over each push button, pop-up help shows you the function of each button.

When you create database areas, storage groups, and databases, the Partition Manager displays them (using various colors, if you have a color monitor) to show which databases occupy extents in which database areas. The Partition Manager also shows whether an extent is used for data or for logs.

You can drag and drop database areas from one storage group to another. This is allowed only for database areas that do not have any extents allocated for any database.

If you try to open the Partition Manager and do not have any partitioned databases on the server, SQLConsole warns you that partitions are not enabled and asks if you

want SQLConsole to enable them for you. If you proceed, SQLConsole changes the value of the *partitions* keyword in the SQL.INI file from 0 (off) to 1 (on).

Partition a database

1. Create one or (typically) more database areas of specified sizes within the file system. This process allocates room in which to create storage groups for your databases.
2. Create a storage group by giving a name to a list of database areas.
3. Create a new database in a storage group.

Read the *SQLBase Database Administrator's Guide* for detailed information about creating and maintaining partitioned databases.

Chapter 7

Managing SQLBase Performance

SQLConsole lets you view server and database activity down to the process, cursor, and lock level. You can use this information, which is presented as easy-to-read graphs and forms, to make optimization decisions and detect problems. This chapter discusses how to view and collect performance information.

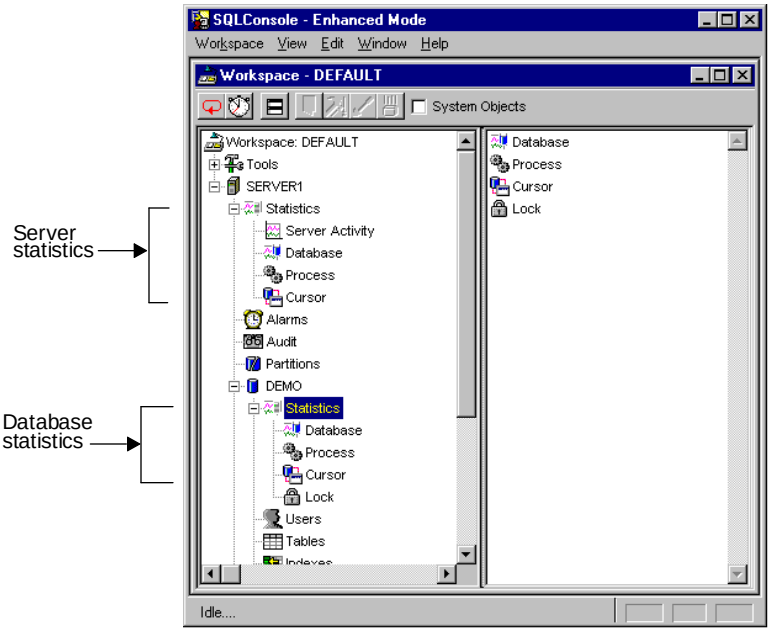
Monitoring your SQLBase server

To monitor your server, begin by observing overall performance with server-level statistics. If you recognize a problem, you can open a specific window to see database, process, cursor, or lock activity. The key statistics are carried from the Server Activity window to the process and cursor windows, so you can isolate a problem to a specific user or cursor.

The Statistics nodes

By default, each workspace contains a server and database Statistics node, and each child node under the Statistics node represents more detailed information. Double-click any statistics node to open an information window. Within most windows, you can display the information in bar chart, pie chart, or line graph format.

The following picture shows the server and database Statistics nodes.



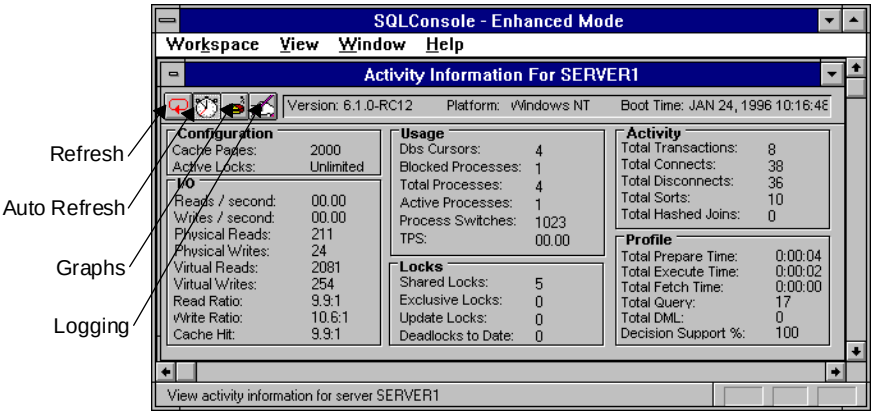
Server connection required

To view database, process, or cursor activity, you must have a server connection. If you need to establish a server connection, read *Connecting a server* on page *6-2*.

Using push buttons

The push buttons in a statistics window let you view graphs, log statistics, and control the refresh rate. To identify a push button, position your cursor over it and wait a few seconds for its label to appear.

The following picture identifies the push buttons in a sample statistics window.



Controlling a window's refresh rate

Information in a statistics window refreshes automatically, unless automatic refresh is turned off. You can turn automatic refresh off through the SQLConsole Settings dialog box (as described in *SQLConsole Settings* on page *5-8*). Alternatively, you can turn off automatic refresh for a single window by clicking the **Automatic Refresh** push button, which depresses the push button (puts it in the "up" position).

Displaying graphs

You can view activity information as a graph by clicking the **Graphs** push button or using the **View** menu item. You can display most values as a bar graph, pie chart, or in other formats.

Writing statistics to log files

To use the statistics information most effectively, log the information to the SQLCON database or to files using the Log Manager. This allows you to examine your system's performance over time and objectively measure the results of any tuning changes you make.

For more information on setting up log files to record statistics, refer to *Logging information statistics* on page *7-13*.

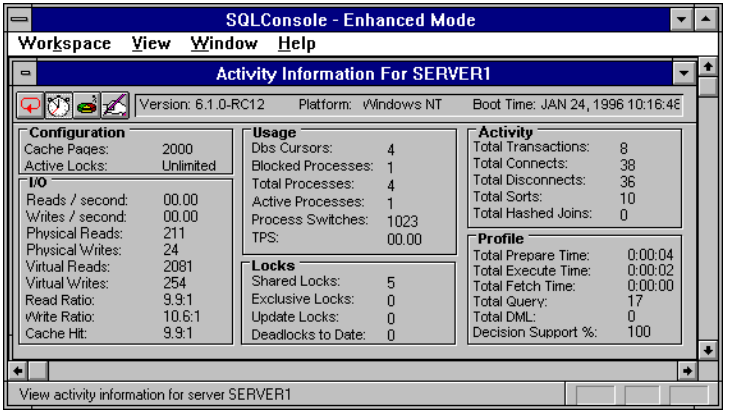
Important statistics to monitor

There are many indicators in SQLConsole that you can use to measure your system's operation, but the following three are some of the most important:

- Blocked processes.
- The cache hit ratio. Work to make this value as high as possible. The best value you can achieve depends on your database and transaction design.
- Disk I/O (input/output) and process switches. A process switch occurs every time SQLBase switches from one process to another. A very high rate of process switching combined with a low physical and virtual I/O indicates that SQLBase is CPU bound.

  A high rate of physical and virtual I/O relative to process switching indicates that SQLBase is I/O bound.

These indicators are tracked on the Activity Information window. This window provides a global picture of the database server. In addition to the three indicators listed above, it contains several other statistics that help determine the overall performance of the server. To open this window, double-click the Server Activity node (under the Server and Statistics nodes on a workspace).



To identify general trends for server statistics, you can log the information (as described in *Logging information statistics* on page *7-13*), use the Alarm Manager (read *Chapter 9, Monitoring Server Conditions with the Alarm Manager*), or simply examine the Activity Information window.

Cache tuning

The Activity Information window displays three cache ratios you may want to track:

- The read ratio represents the number of times SQLBase read a page from the cache for every page it physically moved.
- The write ratio represents the number of times SQLBase wrote a page to the cache for every page it physically moved.
- The cache hit represents the combined read and write ratio. You want the cache hit ratio to be as high as possible. The actual values depend on your system.

These values are cumulative, beginning from the time the server is started or restarted.

Tune the cache

We recommend the following procedure to tune your cache.

1. Initially, select a small cache setting.
2. Set the cache size.

Note: Cache size is one of the server properties you can change by right-clicking a server node and selecting **Properties**. For more information, read *Appendix A, Server and Database Properties*.

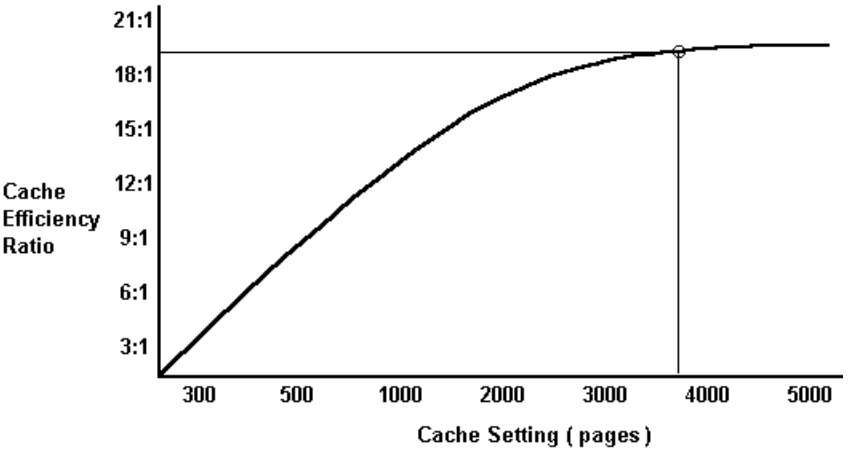
---

3. Run the server for a predetermined period of time either during normal operation or using test scripts. Using test scripts ensures that resulting cache hit ratios are independent of other operational variables. Monitor the cache efficiency and record the results. It may be useful to record results in a log file (read *Logging information statistics* on page *7-13*).
4. Incrementally increase the cache size.
5. Repeat steps 2 through 4 until there is little or no cache hit gain for the increased size.
6. Plot cache hit versus cache size. Select the most efficient cache size.

To select the appropriate cache setting, you must consider the amount of memory available to the server. Ideally the largest cache hit returns the best performance. However, a balance between available memory and cache hit is appropriate.

In general, select the cache setting at which there is little gain in increasing the number of pages. as illustrated in the following graph.

Note that, because of cache management overhead, it is also possible for performance to decrease if you allocate too much cache.



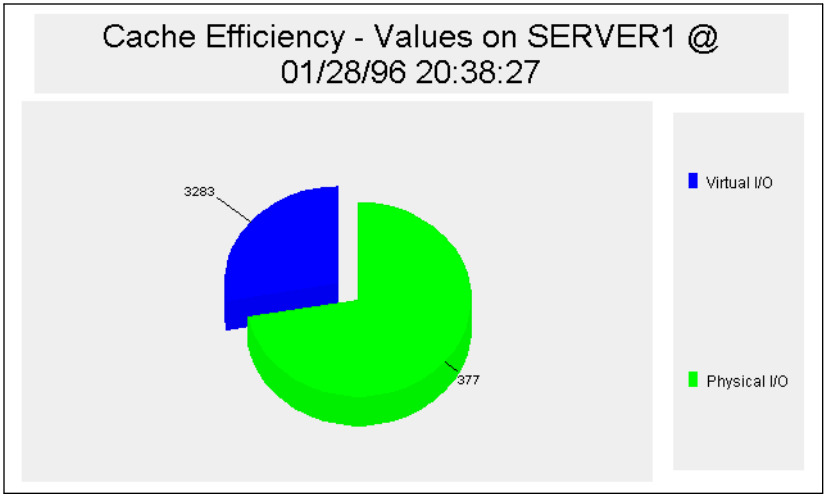
Server activity graphs

From the Activity Information window, you can view an activity graph by clicking the **Graph** push button or using the **View** menu. The available graphs are:

- Cache efficiency.
- Cursor use.
- Execution profile (the amount of time SQLBase spent in PREPARE, EXECUTE, and FETCH states since startup).
- SQL profile.
- SQLBase I/O (input/output).
- Process use.
- TPS (transactions per second).

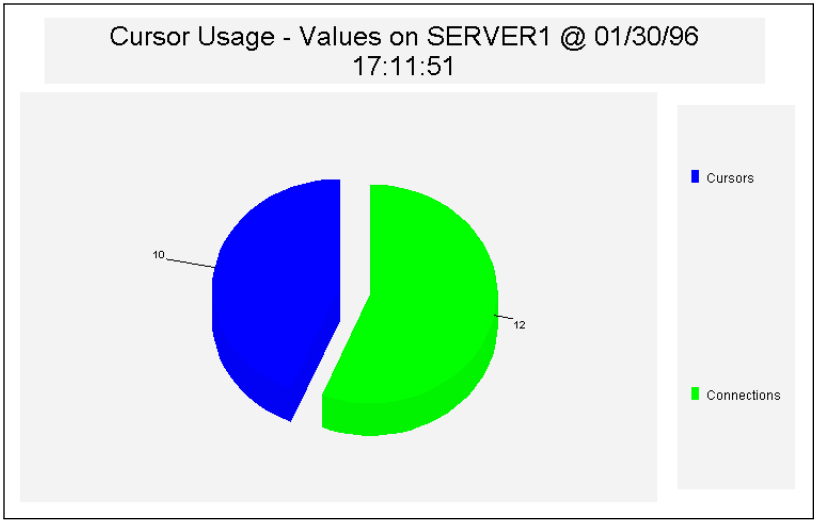
An example of each graph follows. After you display a graph, you can alter it with the **View** menu or the push buttons above the graph.

Cache efficiency graph



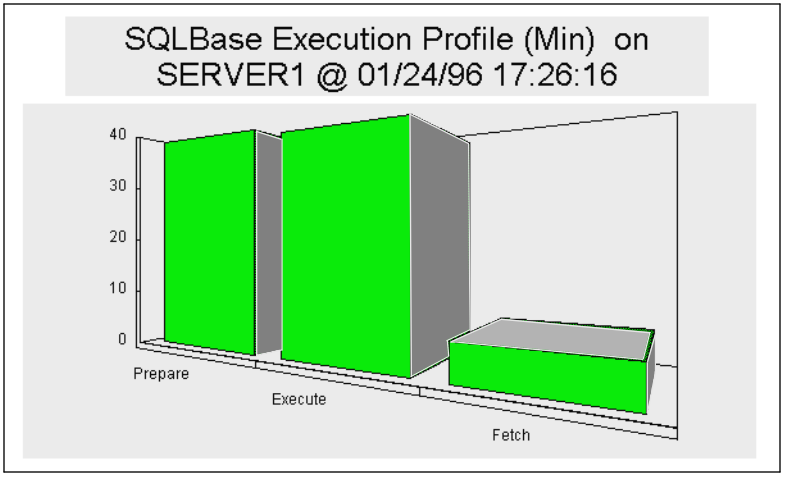
This graph shows the virtual I/O and the physical I/O activity. Larger virtual I/O values indicate more effective use of the cache.

Cursor use graph



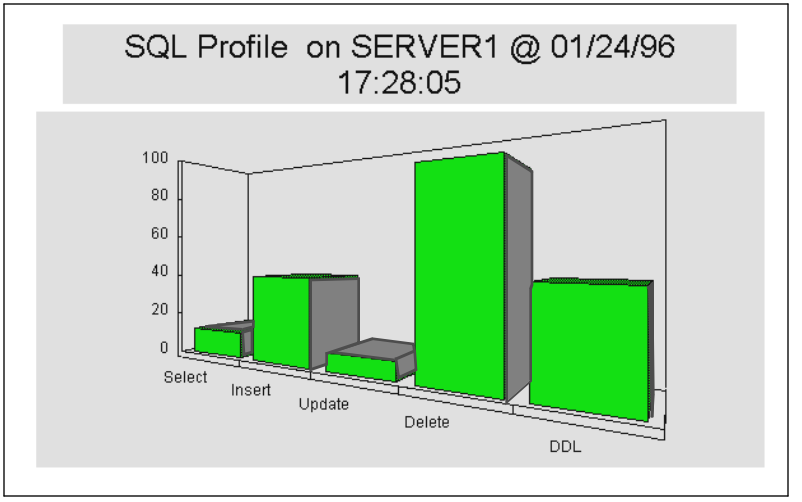
This graph shows the number of cursors versus connections. Cursor use helps you evaluate database performance. Too many cursors per connection consumes resources, while only a few cursors per connection may indicate an application that fails to fully exploit the server.

Execution profile graph



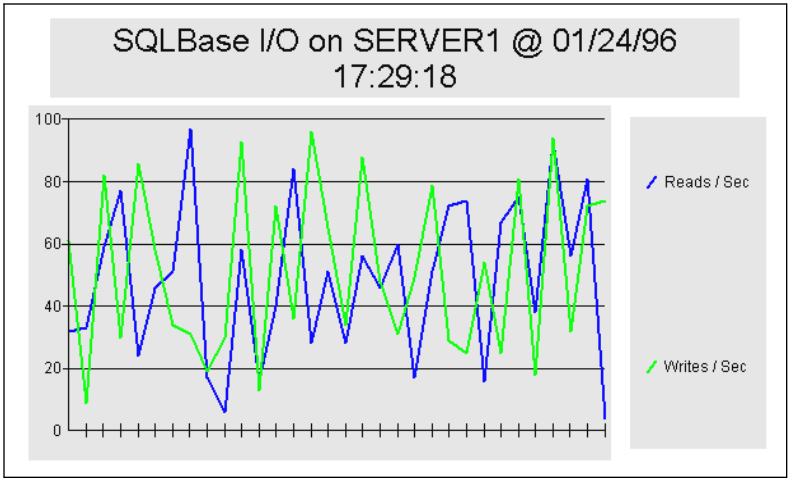
When you optimize a SQLBase server, one of the most important statistics to consider is the time allocated by the server. This graph shows the amount of time in prepare, execute, and fetch states since startup.

SQL profile graph



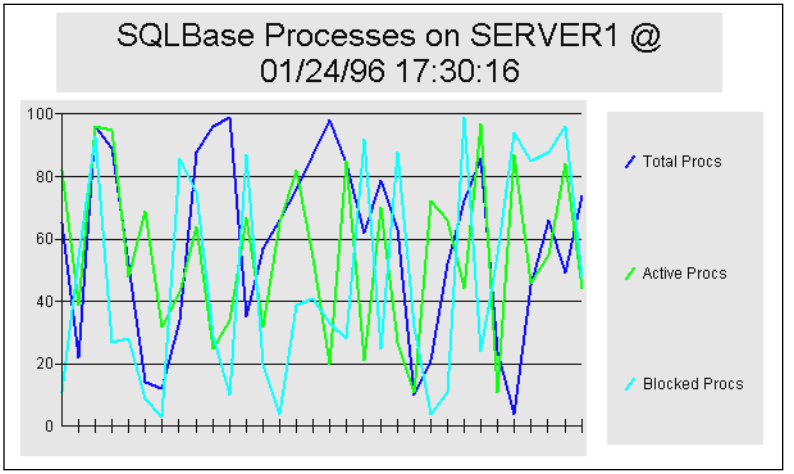
The SQL profile graph shows the types of statements (SELECT, UPDATE, DELETE, INSERT, and DDL) executed on the SQLBase server since startup. This information helps you benchmark or optimize a database server.

SQLBase I/O graph



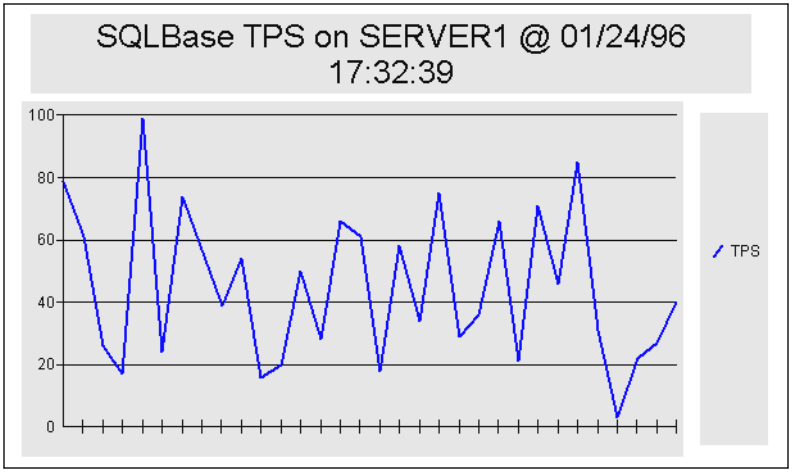
The SQLBase I/O graph is a sliding window that shows the I/O of the SQLBase server. Reads are plotted against writes.

SQLBase processes graph



The process use graph is a sliding window that shows the total number of processes and their state. The graph shows total processes, blocked processes, and active processes. This is one of the most important graphs for measuring the overall state of the SQLBase server.

TPS Graph

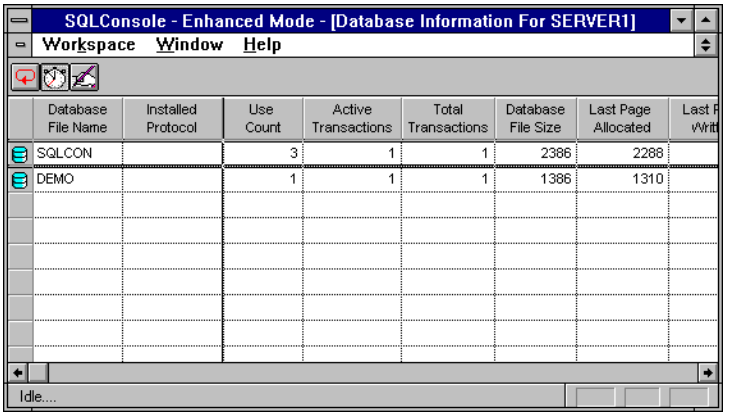


This graph is a sliding window that shows the actual transactions per second for a SQLBase server.

Database statistics

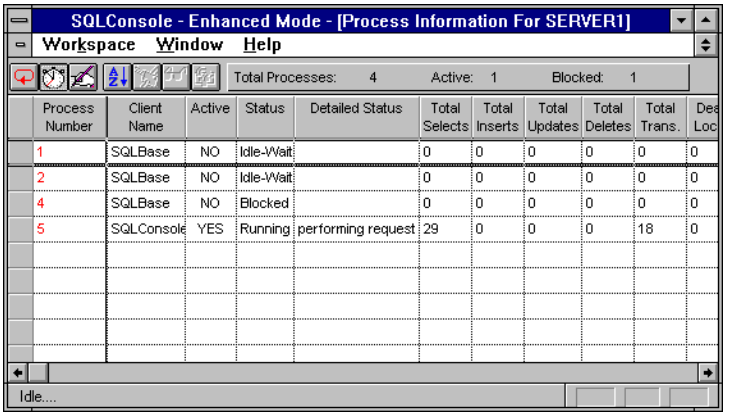
SQLBase servers often have many databases on one server. You can view statistics about all the databases on a server by double-clicking to open a server node, then the Statistics node, and then the Database node. Alternatively, you can view statistics about a particular database by double-clicking to open a database node and then the Statistics node under it. The statistics show which database is receiving the most (and least) use.

The database information window for a server lists each database to which you connected since the server was last started.



Process statistics

Opening the Process statistics node shows you detailed information for each process or open session on the server.



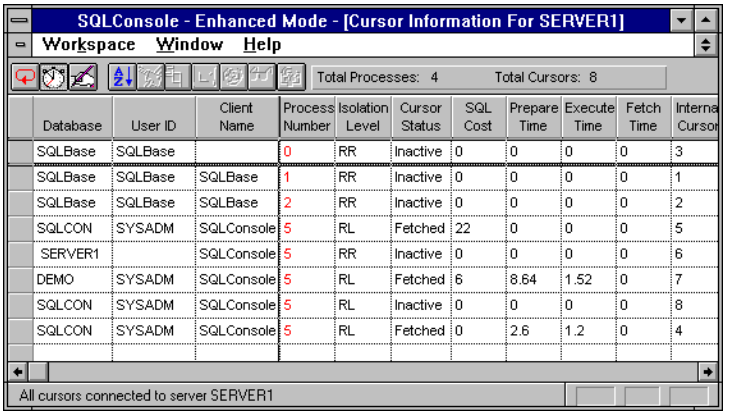
Cursor statistics

The Cursor Information window presents the most detailed view of database server statistics. In general, you use this information to isolate activity that you first observe at the server level. Use the following strategy:

- First, look for general trends in server activity (the Activity Information window).

- If you need more information, use the process information (open the Process node under the Statistics node) to determine which process or processes are responsible for a trend.
- If you need even more information, use the cursor information (open the Cursor node) to locate the cursor or cursors responsible for a trend.

Often problems with a database are the result of improper cursor use by a frontend client application. For example, assume that 100 cursors are connected to a database using release locks isolation level, but one of the cursors is mistakenly using repeatable read. This one cursor leaves locks that can cause concurrency problems. In this situation, you could investigate the problem by using the Cursor Information window to examine the isolation level for each cursor.



Database	User ID	Client Name	Process Number	Isolation Level	Cursor Status	SQL Cost	Prepare Time	Execute Time	Fetch Time	Interna Cursor
SQLBase	SQLBase		0	RR	Inactive	0	0	0	0	3
SQLBase	SQLBase	SQLBase	1	RR	Inactive	0	0	0	0	1
SQLBase	SQLBase	SQLBase	2	RR	Inactive	0	0	0	0	2
SQLCON	SYSADM	SQLConsole	5	RL	Fetched	22	0	0	0	5
SERVER1		SQLConsole	5	RR	Inactive	0	0	0	0	6
DEMO	SYSADM	SQLConsole	5	RL	Fetched	6	8.64	1.52	0	7
SQLCON	SYSADM	SQLConsole	5	RL	Inactive	0	0	0	0	8
SQLCON	SYSADM	SQLConsole	5	RL	Fetched	0	2.6	1.2	0	4

Lock statistics

The Lock Information window shows the locks held by database processes against a database. You access this window by opening the Lock node (under the Statistics node for a database).

The Lock Information window arranges the database objects horizontally in alphabetical order. The database processes are arranged vertically in ascending order of process number. You can check the System Objects or Indexes check box to make these objects appear in the window.

The lock table displays the type of lock held against a table:

- S — Shared (or *read*) lock.
- X — Exclusive (or *write*) lock.
- U — Update (or *intent to update*) lock.
- I — Incremental lock.

- TS — Temporary shared lock.
- TX — Temporary update lock.

On a color monitor, exclusive locks are blue and update locks are red. Double-clicking a row header creates a window that displays all the locks that specific process holds on the database. Double-clicking a column header creates a window that shows all the locks held against a database object.

You can abort a database process by selecting a process row and then clicking the **Disconnect** push button.

Checking for uncommitted transactions

When a process leaves exclusive locks on a database object, no other process can select or alter that data until the transaction is complete. This can create performance problems, and is usually the result of a frontend application failing to commit its transactions. Identifying and correcting this problem is extremely important, especially in multi-user environments.

Logging information statistics

We recommend that you log server and database statistics. This lets you run the server for a predetermined period of time either during normal operation or using test scripts. You can then make changes to the system, use the log file to measure the effect of the changes, and eventually find the optimum configuration for your system.

You can log statistics either to the SQLCON database or to a log file. (Note that the log files used for logging statistics are different from the log files SQLBase uses for backup and recovery.) We recommend logging to the SQLCON database, which lets you issue SQL commands against the statistics you gather.

Enable refresh

Note that in order to log statistics, you *must* enable refresh for the statistics window. You do this any of the followings ways:

- Enable automatic refresh for all statistics windows (read the refresh information in *SQLConsole Settings* on page *5-8*).
- Use the **Refresh** push button (the arrow shaped like an oval) to manually refresh the activity window.
- Use the **Auto Refresh** push button (the clock) to enable automatic refresh for the activity window.

The statistics are logged at the current refresh rate (or the rate at which you manually refresh the window).

How SQLConsole manages log files

Log files represent one calendar day of activity. When a new day of logging begins, the current files are moved to the backup directory and new files are created. Any logging information that occurs in a calendar day is appended to the current log file.

SQLConsole creates log files on a daily basis. If a new session with SQLConsole is started on the same day as the last, the new session's log information appends to today's file. If the new session starts one or more days later, the last set of log files is copied to the backup log file directory, and a new log is created. If a single logging session continues over the period of one day, new logs are automatically created at the start of the new day.

Backup files are overwritten, so if you want to keep data from more than two days, copy the backup logs to another media or move them to a different directory.

Log file names

The name of a log files is based on the name of the connected server at the time of the log file's creation. The file extension is determined by the information window:

- .ACT — Server Activity.
- .PRC — Process.
- .CUR — Cursor.
- .DAT — Database.

For example, SERVER19.ACT is a sample log file name.

Log file updates occur at the current refresh rate. If automatic refresh is off but the logging facility is active, then log file updates happen only when you manually refresh an information window. Data appears in the log files in the same tabular format as displayed in the information windows.

When are log files active?

Once logging is activated, you write data to the log files only if all the following are true:

- You have a server connection.
- The information windows you want to log are open.
- You refresh the data in the information windows in some way (see *Enable refresh* on page *7-13*).

How to log information statistics

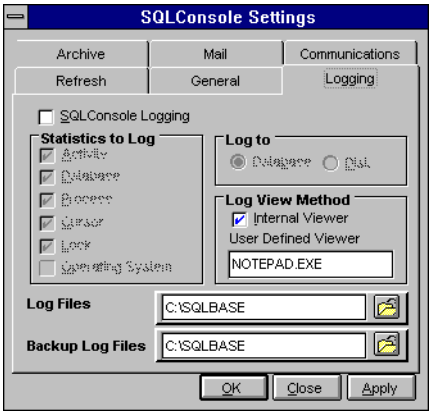
Logging information involves three steps:

- Open the SQLConsole Settings node to enable logging and specify logging settings (whether to log to the SQLCON database or to disk, for example).
- Open one or more statistics information windows (under the Statistics node for a server or database). You can open and capture statistics from any combination—all, none, or a subset—of activity windows. Logging stops when you close an information window.
- View the information statistics and turn off logging.

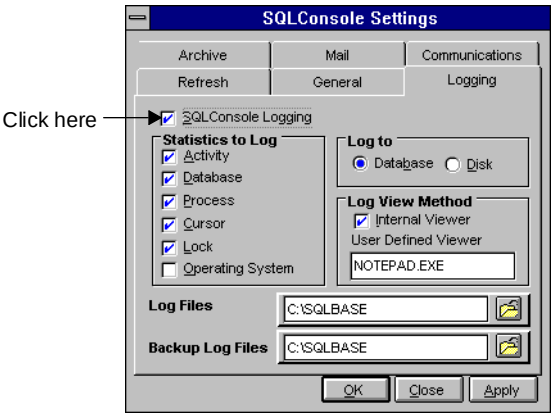
These steps are explained below.

Specify logging settings

1. On a workspace, double-click the SQLConsole Settings node (under the Tools node) to open it.
2. Click the **Refresh** tab, then ensure that **Automatic Refresh** is turned on.
3. Click the **Logging** tab.



4. Click to put a check in the SQLConsole Logging check box.



5. In the Statistics to Log group box, put a check on each statistic you want to log (or accept the defaults).

Note: After you open a statistics information window you can disable logging in specific information windows by clicking to depress (put in the "up" position) the **Logging** push button on the window's toolbar. The **Logging** push button is the push button with a hand on it.

---

6. In the Log to group box, select whether to log to database or disk.

   Logging to a database logs the information to the local SQLCON database. No further configuring is necessary for logging to database.

7. In the Log View Method group box, specify how you want to view the log files.
8. If you selected Log to Disk in step 6, select the location for the log files. You can use the **Browse** push button to view your directory structure.
9. In the Backup Log Files field, select the location for the archive log files.
10. Click **OK**.

Open a statistics window

In order to start logging, you must open the corresponding statistics window for each statistic you selected in step 5 in the preceding procedure.

Logging stops when you close the corresponding information window.

Viewing SQLConsole log files

One of the logging options you set (see *Specify logging settings* on page *7-15*) is the method for viewing log files. By default, SQLConsole's own viewer is active. It can view files up to 6.5 megabytes in size, which is sufficient for most situations.

If you want, you can use your own viewer. Use a Windows editor if possible. If you use a DOS editor, turn automatic refresh off before opening a log file. The log files are created in ASCII format, and some lines may be quite long. Therefore, select an editor that does not word-wrap the lines.

You can view the log files at any time.

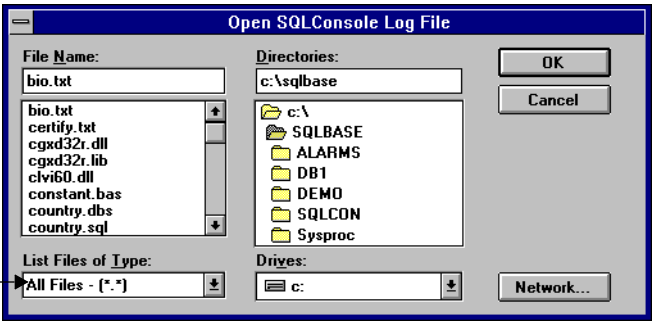
View a log file logged to disk

To use this procedure, you must first write information to disk (see step 6 of *Specify logging settings* on page *7-15*).

1. Double-click the Log Manager.

   The Open SQLConsole Log File dialog box appears.

Use this list box to help you find a log file. For example, you can list only



2. Enter a file in the File Name field and click **OK** to view the log.

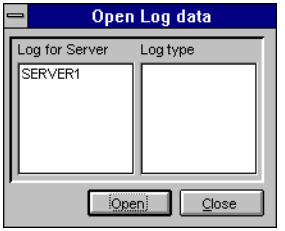
   The log file contains the name of the server followed by an identifying extension. For more information, read *How SQLConsole manages log files* on page *7-14*.

View a log file logged to database

To use this procedure, you must first write information to the SQLCON database (see step 6 of *Specify logging settings* on page *7-15*).

1. Double-click the Log Manager node (under the Tools node on a workspace).

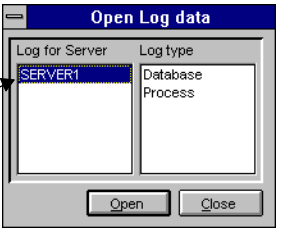
The Open Log Manager dialog box appears.



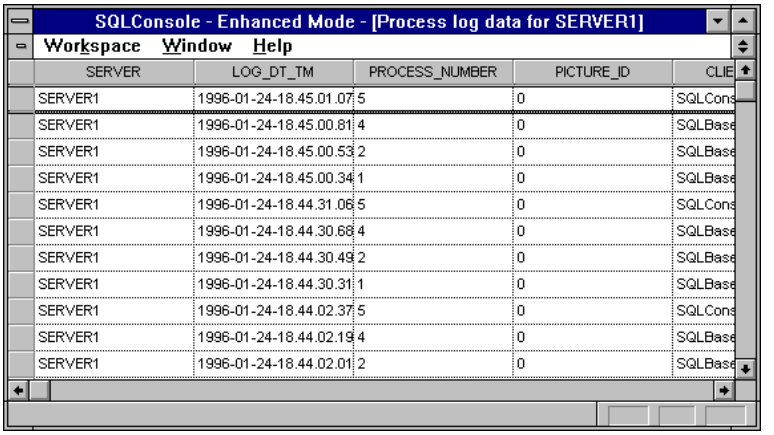
2. Double-click on the server in the Log for Server window.

   The statistics you are logging appear in the Log Type window.

Click the server to make the log types appear.



3. Double-click on a log type (for example, Process) to open it. For example, in the picture above, if you double-click the Process log type the following window opens.



The Open Log Manager dialog box remains open so you can open multiple log type windows.

Turning off logging

Once you specify logging settings and open a statistics window, SQLConsole continues to log statistics until you close the window or turn off logging.

To turn off logging, double-click the SQLConsole Settings node (under the Tools node on a workspace), click the **Logging** tab, and then click to remove the check in the SQLConsole Logging check box.

Remember to turn off logging when you finish. Depending on your server and the refresh rate, the log files can grow to a large size. Cursor logs, the largest, can grow to approximately six megabytes over a 24 hour period, refreshing once a minute. SQLConsole watches how much disk space is available and automatically shuts down logging when it finds only one megabyte left.

Chapter 8

Server Auditing

This chapter describes the SQLBase server auditing feature and show how to use it from SQLConsole.

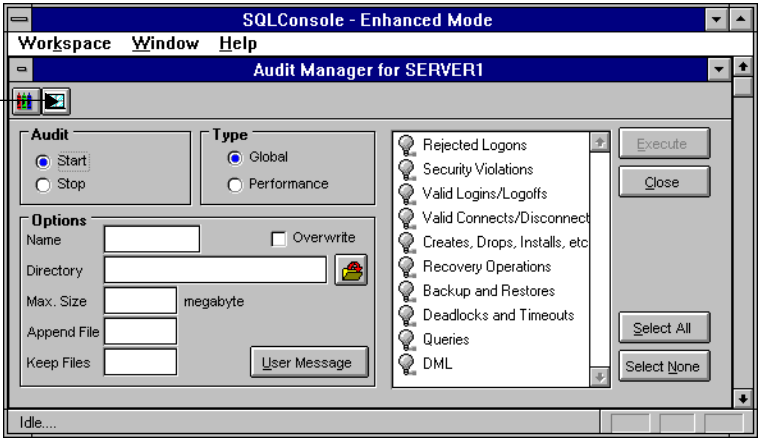
For additional information, refer to the auditing discussion in the *SQLBase Database Administrator's Guide*.

Server auditing

Server auditing lets you audit server activities and performance. For example, you can monitor who logs on to a database, which tables they access, and if they attempt to access data for which they do not have privileges. You can also measure server performance by gathering information on command execution time and long-running transactions.

To perform server auditing from SQLConsole, use the Audit Manager. You open the Audit Manager by double-clicking the Audit node (under a server node) on a SQLConsole workspace.

Audit Viewer



To gather server auditing data, you start one or more ***audit operations*** that write output to an associated ***audit file***.

You can have up to 32 active audit operations running concurrently. However, generally you do not need more than one or two, since you can record different types of information within each audit. Also, be aware that each additional audit operation can affect performance.

All concurrent active audits must have unique names.

Audit file

An audit operation writes its output to an ***audit file***. This is a simple ASCII file that you can print or access through either an editor or SQLConsole (using the **Audit Viewer** push button pictured above).

The audit file collects various types of system and performance information depending on the ***audit type*** and ***category*** you choose. You can also direct a message to an audit file to document a system activity.

How SQLConsole names and manages audit files

An audit file name has the format AUDITNAME.X, where ***X*** is an ascending value. For example, starting an audit operation with the name MYAUDIT generates an output file called MYAUDIT.1

You can record information to one audit file of unlimited size, or to a series of sequential files that have a specified size. When a file reaches this defined size, SQLBase automatically generates a new file. For example, if you specify a maximum size of 100 kilobytes when you start the MYAUDIT audit, SQLBase automatically creates and starts writing output to MYAUDIT.2 when MYAUDIT.1 reaches a size of 100 kilobytes.

Maximum number of audit files

Since the sequence of audit files is controlled by the file extension, you can theoretically have up to 1000 audit files for each individual audit (though this may not be practical). The audit file extension sequence ranges from AUDITNAME.1 to AUDITNAME.999. AfterAUDITNAME.999, the next file is called AUDITNAME.0, and then wraps around to begin the sequence again with AUDITNAME.1.

You can automatically delete old audit files by telling SQLBase to keep only a certain number of files besides the current file. For example, if you specify a KEEP value of 2 for an audit called MYAUDIT, SQLBase automatically deletes ***MYAUDIT.1*** when it creates ***MYAUDIT.4***, since it can keep only two old files ***(MYAUDIT.2*** and ***MYAUDIT.3***).

You set the KEEP value in step 10 of the procedure *Start an audit* on page *8-5*.

Starting and stopping an audit

For information on starting an audit, read *Start an audit* on page *8-5*.

When you click **Execute** to begin the audit, SQLBase creates the following entry in the appropriate section of your configuration file (SQL.INI):

```
AUDIT=[type],auditname,[directoryname],[size-integer],
[append-integer],[keep-integer],[OVERWRITE],x,...
```

For example:

```
AUDIT=GLOBAL,SECURITY,C:\SQLBASE,1000,1,1,OVERWRITE,1,23
```

For descriptions of these parameters, refer to the START AUDIT command documentation in the *SQL Language Reference*. The ***x*** field represents an information category. For more information, refer to the auditing section of the *SQLBase Database Administrator's Guide*.

An audit remains active while the server is running (even if you shut down SQLConsole). It stops when you shut down the SQLBase server, but restarts when

you bring the server back up. To stop an audit operation, open the Audit Manager and stop the audit (as described in *Stop an audit* on page *8-6*).

Viewing the audit file

You can view an audit file with the Audit Viewer push button (see the picture on *Server auditing* on page *8-2*).

Types of audit operations

SQLBase supports two types of audit operations: ***global*** and ***performance***. These two audit types record different types of information. You specify what type of information to record by indicating its ***category*** when you start an audit.

Both audit types automatically record the following general information in the audit file:

- The start of an audit.
- Any user-defined message that was issued to the audit file. Read *Write a user-defined message to the audit file* on page *8-6* for more information.
- The end of an audit.

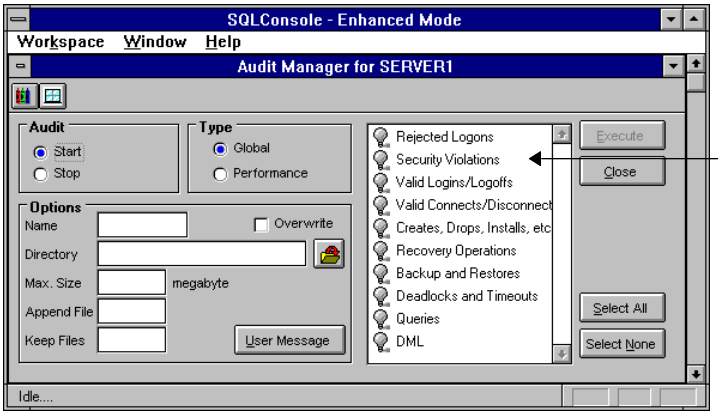
For detailed information about global and performance audit types, audit file examples, and information on how to read the audit file, refer to the auditing section of the *SQLBase Database Administrator's Guide*.

Using the Audit Manager

This section tells how to perform common tasks in the Audit Manager.

Start an audit

1. On your SQLConsole workspace, double-click the Audit node (under a server node) to open the Audit Manager dialog box.

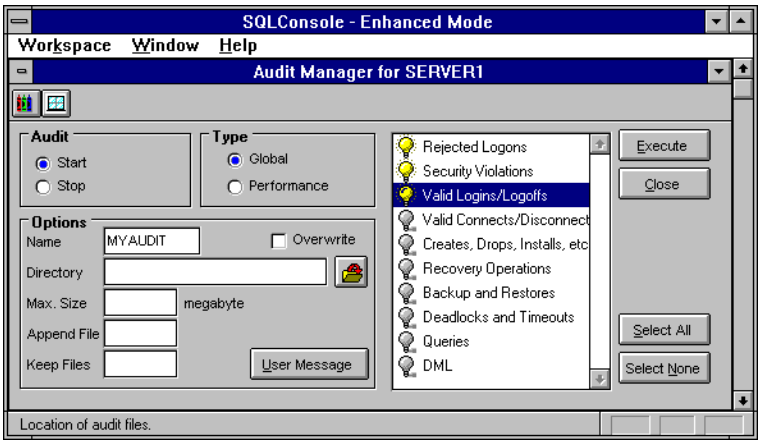


2. Click **Start**.

3. In the Type group box, click either Global or Performance. Depending on the audit type you choose, the list box on the right displays the corresponding data the audit can track.

4. Double-click any entry or entries in the right list box to turn it on (yellow) or off (gray). All items with a lit light bulb are included in the audit.

5. In the Name field, enter the audit name. For example, enter `MYAUDIT` in the Name field.



SQLBase can have as many as 32 active audits, and each must have a unique name. The audit name is the prefix of the audit file.

6. In the Directory field, enter a directory where the audit file will reside.

   If you want, use the **Browse** push button to help you locate files and directories.

7. If the audit name is the same as existing audit files in the selected directory, click the Overwrite field to overwrite the old audit files.

8. If you want to continue where a previous audit left off, enter the file name and extension in the Append File field.

9. Enter the maximum audit file size in megabytes in the Max Size field.

   When the size of the file reaches this number, the current file will be closed and another file will begin.

10. In the Keep Files field, enter the number of audit files for SQLBase to keep. For more information, refer to *Maximum number of audit files* on page *8-3*.

11. Click **Execute** to begin the audit.

Stop an audit

1. On your SQLConsole workspace, double-click the Audit node (which is located under a server node) to open the Audit Manager dialog box.

2. Click **Stop**.

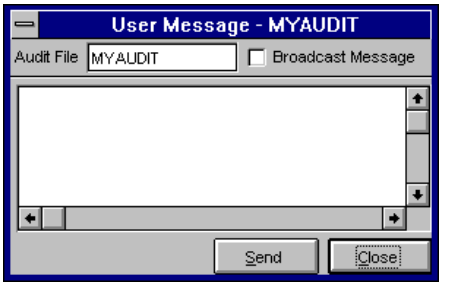
3. In the Name field (in the Options group box), enter the name of the audit to stop, or leave the Name field blank to stop all audits.

4. Click **Execute** to stop the audit or audits.

Write a user-defined message to the audit file

1. Click the **User Message** push button in the Audit Manager dialog box.

   The User Message dialog box appears.



2. Enter the name of the audit file to which to write the message, or click **Broadcast Message** to write the message to all active audits.
3. Enter the text of the message in the text field. The message can contain up to 254 characters.
4. Click **Send**.

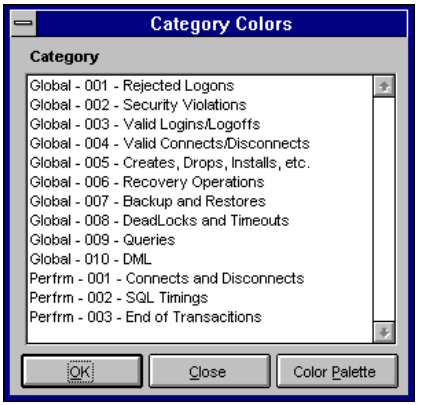
Note: If you do not know the name of an audit that is running and you want to stop it, you must leave the NAME field blank and thus stop all audits. SQLBase is unable to show you the names of the active audits.

Set category colors

The **Color** push button on the tool bar of the Audit Manager allows you to define a color for a given category of audit data. The color settings apply to the category list box in the Audit Manager and to the Audit Viewer.

1. Click the **Color** push button on the Audit Manager tool bar.

   The Category Colors dialog box appears.



2. Click one or more categories, then click **Color Palette**.
3. In the Color Palette dialog box, click the color you want, then click **OK**.
4. When your color selections are complete, click **OK** to exit the Category Colors dialog box.

Chapter 9

Monitoring Server Conditions with the Alarm Manager

The Alarm Manager is a flexible way to automate responses to a variety of server conditions. This chapter describes how to use the Alarm Manager.

What is the Alarm Manager?

To help you circumvent potential problems, SQLConsole's Alarm Manager alerts you when user-defined thresholds for various server conditions, or *alarms*, are exceeded. An alarm can trigger an action, such as killing the connection that is causing the alarm condition.

Examples of alarms include:

- An excessive number of inactive processes.
- An excessive number of exclusive locks.
- An excessively low cache hit ratio.

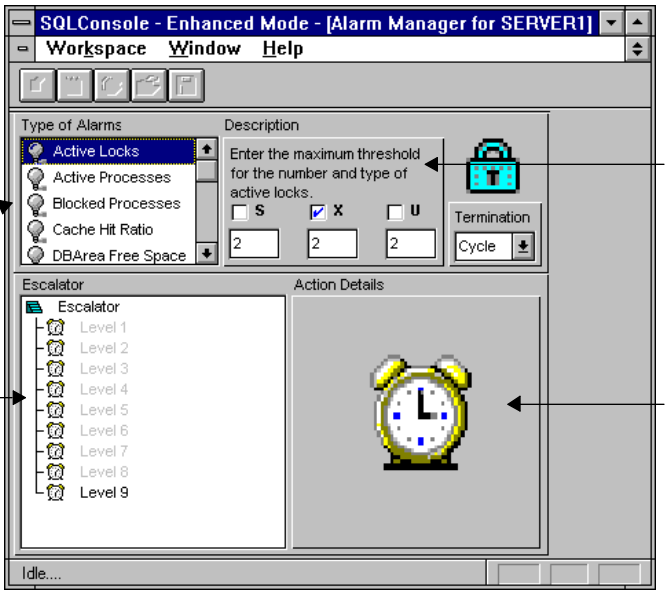
The behavior that occurs when an alarm condition is met can be simple, such as a flashing window or a beep on the SQLConsole workstation. You can also define more powerful behavior, such as executing a script.

The Alarm Manager interface

You access the Alarm Manager by double-clicking the Alarm node under a server node on a SQLConsole workspace.

Note: You cannot use the Alarm Manager (and it does not appear on your workspace) if you are running in standard mode.

The following sections describe the parts of the Alarm Manager window.



Alarm types

An alarm is a particular server condition. You can set any of the pre-defined alarms listed in the following table.

Alarm Type	Description
Active locks	Triggered when the number of one or more lock types equals or is greater than the threshold you set. The lock types you can set are: • S — Shared lock. • X — Exclusive lock. • U — Update lock.
Active processes	Triggered when the number of active processes equals or is greater than the threshold you set.
Blocked processes	Triggered when the number of blocked processes equals or is greater than the threshold you set.
Cache hit ratio	Triggered when the percentage of cache hits equals or is less than the threshold you set.

Alarm Type	Description
Database area free space	Triggered when the percentage of DBAREA free space equals or is less than the threshold you set.
Inactive processes	Triggered when the number of minutes a process remains inactive meets or exceeds the threshold you set.
Isolation level	Triggered by the existence of a cursor using one of the following isolation levels, depending on which are checked: • CS — Cursor stability. • RR — Read repeatability. • RO — Read only. • RL — Release locks.
Scheduler	Triggered when the Scheduling Manager detects an error. This alarm has only one level of escalation.
Server down	Triggered when the server goes down.
Storage group free space	Triggered when the percentage of STOGROUP free space equals or is less than the threshold you set.
Total cursors	Triggered when the number of total connected cursors equals or is greater than the threshold you set.
Total processes	Triggered when the number of total connected processes equals or is greater than the threshold you set.
CPU utilization	The maximum percent threshold for the CPU utilization.
Disk read bytes per second	The maximum threshold for the disk read bytes/second.
Disk write bytes per second	The maximum threshold for the disk write bytes/second.
Memory free	The minimum threshold for the memory free.
Network read bytes per second	The maximum threshold for the network received bytes/second.
Network routed bytes per second	The maximum threshold for the network routed bytes/second.

Alarm Type	Description
Network write bytes per second	The maximum threshold for the network transmitted bytes/second.

Escalator

In the Escalator you define the behavior that you want to occur whenever the conditions for an alarm are met or exceeded. For example, if five or more exclusive locks occur on the server (this is an "active locks" alarm), you can make the Alarm Manager beep and flash for two minutes (this is behavior you define in the Escalator).

Each alarm type has nine levels of escalation that you can define, except the Scheduler alarm, which has only one level. Level nine is the *lowest* priority and level one is the *highest*. When an alarm is first triggered, the lowest priority level of escalation is executed. Should an alarm activate and persist for a specified time period, the Alarm Manager escalates the alarm to the next higher active escalation level and performs whatever actions are defined for that level. If the alarm escalates to the highest active level and the alarm still persists, the alarm terminates.

Each level in the Escalator has three attributes:

- Color — The color of the text for the alarm.
- Interval — How long to remain at this level. By default, the interval is set to one minute.
- Actions — The behavior you want to occur.

When you set an alarm for the first time, by default level nine is active and its action is set to beep. This ensures that if you enable an alarm, it will, at minimum, beep to alert you to the alarm condition. You can change this default by clicking and setting the nodes in the Escalator section of the Alarm Manager window.

Termination

Specify an alarm's termination in the Termination section of the Alarm Manager window. *Termination* refers to what happens if the alarm conditions continue to exist when the Escalator reaches the highest level set (level one is the highest possible level; level nine is the lowest possible level):

- Cycle — Causes the Alarm Manager to restart again at the lowest Escalator level and work its way up again through the Escalator levels.
- Exit — Causes the alarm to stop.
- Stay — Allows the alarm to remain at its highest active level, executing its actions every interval thereafter. If the alarm conditions no longer exist, the

Alarm Manager resets itself and awaits the next alarm, starting again from the lowest level.

Escalator actions

For each level in the Escalator, you can enable no action or one or more actions. To enable an action, select it and then click the **Enable** push button (so the switch on the push button moves from "down" to "up.")

The possible actions are defined in the following table.

Action	Description
Flash	Causes SQLConsole's title bar to flash.
Beep	Causes the workstation on which SQLConsole is running to beep.
Burn	If the Events window is minimized, the newspaper icon it uses appears to be on fire.
Snapshot	Causes the database process cursor and lock windows to log the current state of the server.
Disconnect user	For inactive processes alarm only. Selecting this action causes the inactive process to be disconnected.
Mail message	Sends e-mail messages to specified a specified user or users.
Execute script	Executes a script file that you specify. Use the Details push button from the toolbar (the push button that looks like an open file folder) for a file selection dialog.
Page beeper	Pages a person or list of people through their digital pager. Enter the desired information in the Action Details section of the Alarm Manager window.

Mail messages

You can define an alarm to send pre-written e-mail to prearranged users. To do this, use the Mail Message action in the Escalator.

When you select this action, the Alarm Details section of Alarm Manager window shows the available messages. You can send one of these messages or prepare a new one.

Choose one or more messages by selecting it and then clicking the **Check** push button. All messages with a check next to them when you exit the dialog are sent when the alarm reaches this level.

In order to use the mail message action, you need configure e-mail to work with SQLConsole. To do this, configure the Mail tab of the Settings property sheet. For more information, read the e-mail information in *Description of settings* on page *5-9*.

Page beeper

You can define an alarm to call a digital pager if the SQLConsole machine is equipped with a dial-out modem. To do this, use the Page Beeper action in the Escalator.

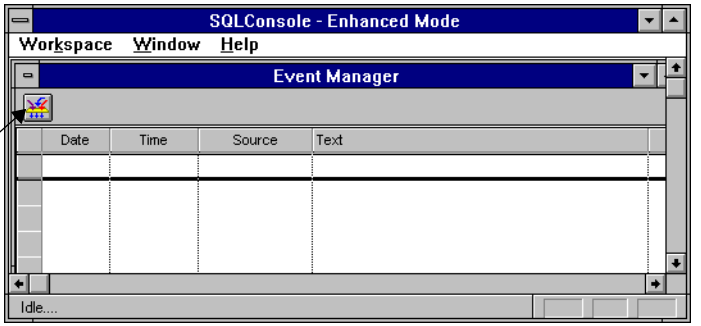
When you select this action, the Action Details window shows you pager contact information. Click the **New** push button to enter a name, phone number, and message. Then double-click the row header to select or deselect the pager, or use the **Select** push button when the line is highlighted. Continue adding or selecting pagers until all desired pagers are entered and selected, then click **Done**.

For an overview of setting SQLConsole to work with your modem, read the communications information in the section *SQLConsole Settings* on page *5-8*.

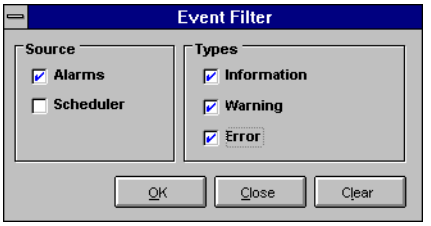
Tracking alarms

Whenever the Alarm Manager (or the Scheduling Manager) logs information, the Event Manager window opens.

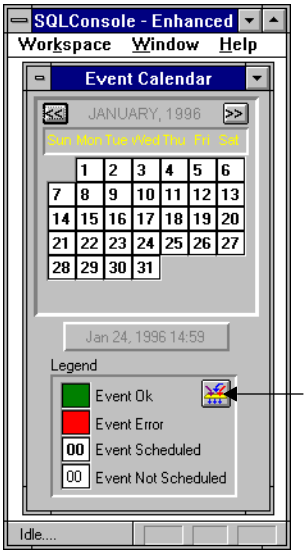
Use the Filter push button to set the information that appears in the Event Manager.



Use the Event Manager's **Filter** push button to set the level of information you want to appear in the Event Manager window.



You can use the Calendar (under the Tools node) to show you a monthly representation of alarm activity, as illustrated in the following picture. Use the Calendar's **Filter** push button to set the level of information you want to appear in the Calendar.



Alarm example

This example sets the Alarm Manager to react to the presence of five exclusive locks and uses the Escalator to execute different alarm actions while the alarm persists.

Select an alarm

1. On a SQLConsole workspace, double-click the Alarm Manager node to open it.
2. In the Types of Alarms list box, double-click Active Locks.

This turns the alarm's lightbulb "on."

Next, set the lock information in the Description window to the right.

3. Click the X (exclusive lock) check box to turn it on. (This may be checked by default.)

   Leave the S and U check boxes blank.

4. For the X lock, enter 5 as the threshold.

5. Set Termination to Cycle. (This may be the default that appears.)

   This causes the alarm to repeat the sequence of actions in the Escalator when it has finished all of them (if the alarm conditions still exist).

   Next, proceed directly to the next procedure to program the behavior you want to occur when the alarm conditions are met.

Program the Escalator

1. Note that Level 8 in the Escalator is grey, indicating it is disabled. Click on Level 8, then click the Enable push button.

   This moves the switch of the Enable push button from down to up, enabling Level 8.

Note: To identify a push button, put your cursor over it. For example, when you put your cursor over the push button with the switch on it, "enable" appears in pop-up help.

---

2. Double-click Level 8 to expand it.

   Three nodes appear under it: Color, Interval, and Actions.

3. Double-click the Color node.

   A color selection dialog box appears.

4. Select red (in the second box down from the far left). This color will be used for alarm log entries.

Note: You can select any color without affecting the operation of the alarm.

---

5. Click **OK** to exit the dialog box.

6. Double-click the Interval node.

7. In the dialog box that appears, change the value from 1 to 2 minutes.

This tells the Alarm Manager to wait two minutes before moving to the next Escalator level.

8. Click **OK**.

   Next tell the alarm what actions to perform.

9. Double-click the Actions node.

   A list of possible actions appears under the Actions node. The actions in grey are disabled for this level.

10. Click Flash, and then click the Enable push button (the push button with a switch).

    The switch on the enable push button moves from "down" to "up," enabling the Flash action.

11. If the Beep action is disabled, enable it by clicking Beep and then clicking the Enable push button.

    Now the Alarm Manager knows to flash and beep when it reaches this level.

12. Repeat this procedure for Level 7, with the following exception: instead of the Flash action (in step 10), choose Snapshot.

This sets an alarm that shows the basic functionality of the Alarm Manager. If five or more exclusive locks occur on the server, the Alarm Manager beeps and flashes for two minutes. After two minutes, it beeps again, and takes a snapshot of the current server activity by logging activity statistics, process, cursor, and lock information. After two more minutes, it cycles and performs the actions again until the alarm condition stops.

An easy way to test this alarm is to create exclusive locks by issuing an UPDATE STATISTICS ON DATABASE command in SQLTalk without committing the transaction.

Chapter 10

Automating Maintenance with the Scheduling Manager

The Scheduling Manager lets you automate many database maintenance tasks. This chapter discusses:

- What the Scheduling Manager does.
- How to use the Scheduling Manager.
- A Scheduling Manager example.

What is the Scheduling Manager?

The Scheduling Manager lets you automate database maintenance tasks, such as database backups.

You access the Scheduling Manager under the Tools node on a SQLConsole workspace. SQLConsole must be running in enhanced mode. The Scheduling Manager is unavailable in standard mode.

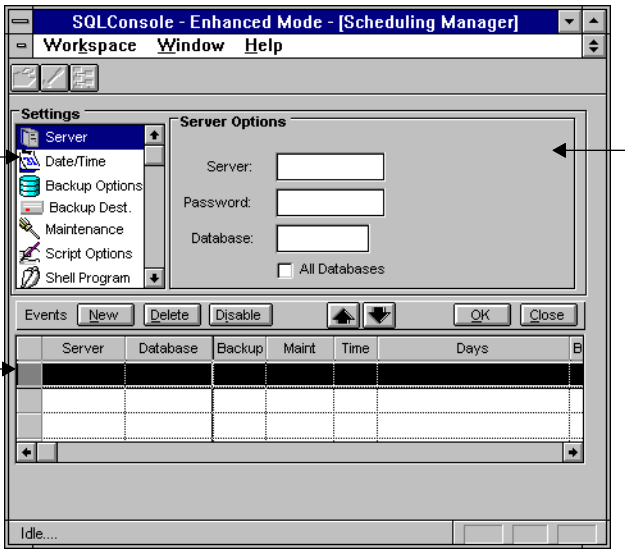
The Scheduling Manager is event-driven

To use the Scheduling Manager, you create one or more events. An event is a group of actions that occur at a specific time. For example, you can create an event that runs the BACKUP SNAPSHOT command at a certain time on a certain database.

Each event is represented by a row in the bottom part of the Scheduling Manager window.

The Settings list box. You define actions for each event here.

The Event window. Each event you create appears here.



The Options group box. You enter detailed information about an action here. These fields change according to what you select in the list box at the left.

Viewing and managing events

You view all existing events in the Event window. When you click on any event in this window, the Options group box shows the information for that event. If you then click an action (in the Settings list box) and enter information in the Options group box, you automatically update the selected event with the new information.

When you click **New** to create a new event, the new event uses the options from the previous event (if any) as defaults. You then customize the new event as needed,

clicking the Settings list box and entering whatever changes you want in the Options group box. This makes it easy to set up several similar events.

Enabling or disabling events

An event can be enabled or disabled. When disabled, the event is not executed as scheduled. This is useful if you want to create events that run only occasionally or irregularly.

An event is enabled when the light bulb in the row header is "lit" and disabled when the light bulb is dark. To disable an event, select its row and then click the **Disable** push button or double-click the row header.

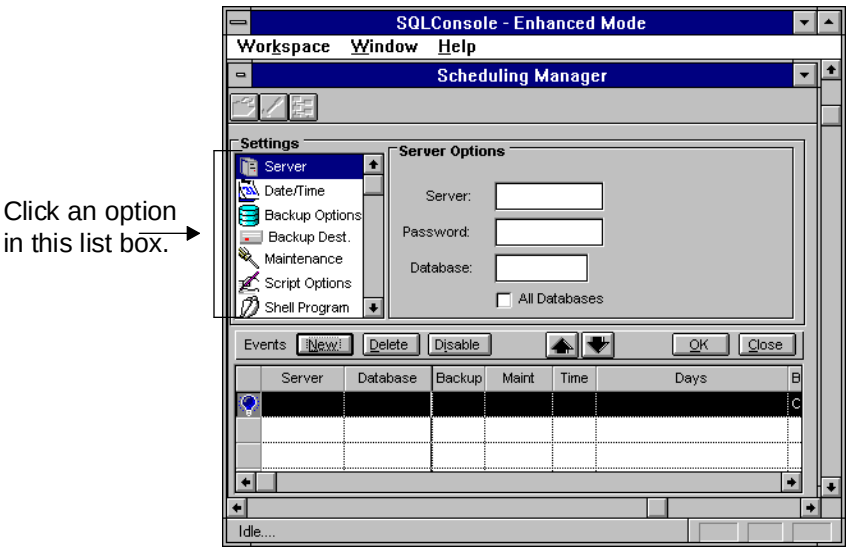
Schedule an event

1. Double-click the Schedule Manager node (under the Tools node on a SQLConsole workspace).

   The Scheduling Manager dialog box opens.

2. Click **New**.

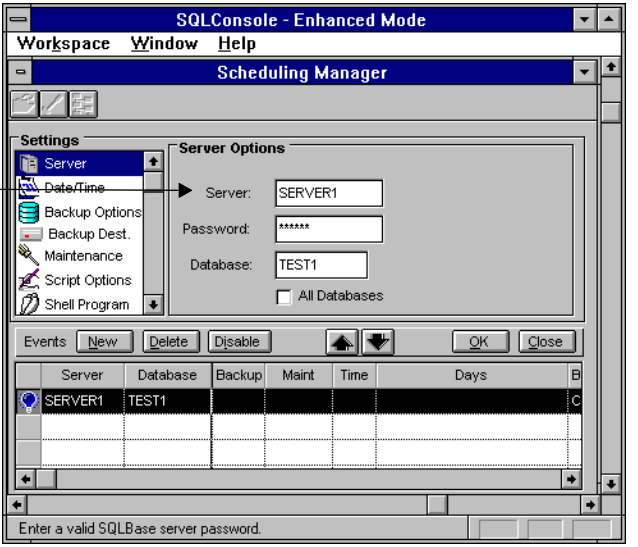
3. Click on an option in the Settings list box (the Server option is selected by default).



4. Enter detailed information about the Action in the Options group box.

For example, enter the server, password, and database information as shown.

Enter detailed information about the option here



SQLConsole automatically enters the information in the lower part of the dialog box.

5. Repeat steps 3 and 4 for each option that applies to your event.

Note: Each new event uses actions from the previous event as defaults. Because of this, you may need to *remove* certain information. For example, if backup information appears by default for your event and you want to remove it, click Backup Options (in the Actions list box) and then click to remove each check in the Backup Options group box.

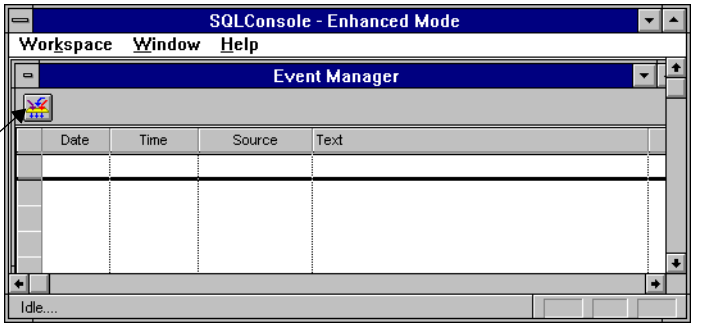
---

6. Click **OK** to save the event and exit the dialog box.

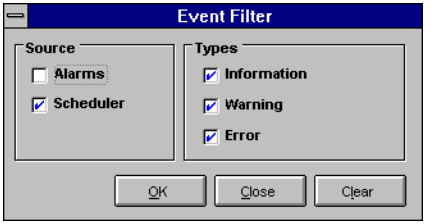
Tracking events

Whenever the Scheduling Manager (or the Alarm Manager) logs information, the Event Manager window opens.

Use the Filter push button to set the information that appears in the Event Manager.

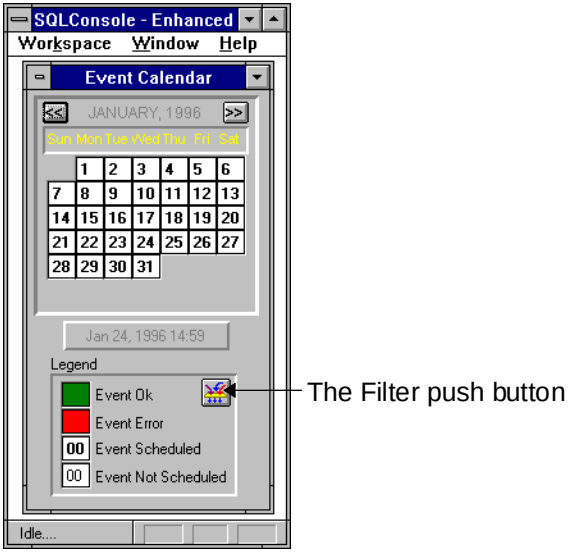


Use the Event Manager's **Filter** push button to set the level of information you want to appear in the Event Manager window.



If you schedule database maintenance events with the Scheduling Manager, you can use the Calendar to see when maintenance events are scheduled to occur, and when they performed successfully. You cannot actually define or schedule maintenance activities with the Calendar; for that, use the Scheduling Manager.

The following picture illustrates the Calendar.



Use the Calendar's **Filter** push button to set which information appears in the Calendar.

Scheduling backups

This section provides detailed information on scheduling backups.

No server connection required

SQLConsole must be running in order to execute a scheduled backup or maintenance event, but you do not need a server connection.

Backup types

It is not possible for one event to include all three types (BACKUP SNAPSHOT, BACKUP LOGS, and BACKUP DATABASE); this would be redundant. More likely, you want to execute a BACKUP SNAPSHOT or BACKUP LOGS command on its own, or a BACKUP DATABASE and BACKUP LOGS in combination.

You may also want to execute a BACKUP SNAPSHOT and BACKUP LOGS together, assuming you have LOGBACKUPS turned on in the database. It makes sense to combine these commands because BACKUP SNAPSHOT does not delete old log files on the server (even though it backs up the log files), and a BACKUP LOGS command *does* delete old log files.

Releasing log files

If you select BACKUP LOGS without also selecting BACKUP SNAPSHOT, a RELEASE LOG check box appears. If this option is checked, a RELEASE LOG command is issued when the event executes. This forces release of the currently active log file, making it available for backup and ensuring the most up-to-date backup. Otherwise, logs are released only when they become full, at which point SQLBase rolls them over and creates a new one. Depending on the size of the log file and the activity in the database, this may be a short or long period of time.

The BACKUP SNAPSHOT command implicitly releases logs but the BACKUP DATABASE or BACKUP LOGS command does not. We recommend that you release logs.

Maintenance options

You may want to perform a CHECK DATABASE or UPDATE STATISTICS command as part of a backup event, or you may want to schedule them separately.

The CHECK DATABASE command checks the integrity of the database. If problems are discovered, it signals whether you should restore a backup of the database or drop a damaged database object.

The UPDATE STATISTICS command updates the index statistics the optimizer uses to weigh the cost of various access paths for a table.

You may want to run these commands often, depending on how active your databases are and how important your data is. Note that refreshing stored commands after an UPDATE STATISTICS is advisable.

Backup destination

These options are determined by the configuration of your workstation, which most likely is a combination of physical drives on the workstation itself and logical drives mapped to a file server.

Select the destination by navigating to it in the File Directory list box. A new subdirectory for each database is automatically created to store the actual database and log file backup files. In the case of backup sets, multiple directories for a single database file are created.

Backup to sets

The Backup To Sets check box (in the Backup Options group box) determines whether the Backup to Sets method is used in the backup event. There are two main advantages to using this method:

- Each instance of a database backup and all the log files associated specifically with it are neatly packaged together.
- More importantly, the possibility of recovering from a corrupted database file is increased.

If Backup To Sets is on, then each time the backup event occurs a new subdirectory is created to hold the backup files. This only applies if the event contains a BACKUP SNAPSHOT or BACKUP DATABASE. Log backups that occur between database backups are written to the most recent set directory.

Using Backup To Sets increases the chance of recovering from a corrupted database file because backups that are written to the same location tend to overwrite the previous backups. If a database becomes corrupted and subsequent backups write to the same location, eventually the archived database backup may corrupt too. Keeping sets of backups improves the chances of finding a database backup that is not corrupted and recovering more data. We recommend that you keep separate instances of database backups between each reorganization of a database.

Log

If you use the Backup To Sets option, ensure that the destination of your log backups is the same as that for the database or snapshot backups. For example, if you decide to do a BACKUP SNAPSHOT for all the databases on SERVER1 every Sunday at 1:00 AM using backup to sets, and the backups are to be written to F:\SQLBASE\BACKUPS, send your daily log backups to the same location, also using backup to sets (assuming you perform BACKUP LOGS each day other than Sunday).

If you do not use Backup To Sets, a Delete check box appears above the Backup To Sets check box. In this case, new database or snapshot backups are always directed to the same destination directory. With each new database backup, the existing set of log files becomes redundant. A check mark in this check box means SQLConsole deletes the old log files for you when a new database or snapshot is performed. This helps preserve disk space. If you need to keep the old log files, do not put a check in this check box.

The Overwrite check box determines whether previous backups of a database can be overwritten. If on, the answer is yes. If Backup to Sets is on, this setting has no consequences. Therefore, you would most likely have the Overwrite option on if you are backing up to one location and you do not care about overwriting the old backup. Typically, if Backup to Sets is not selected, the overwrite checkbox is off. (This assumes that for your backup procedure, after an online backup is performed, contents of the backup directory are moved to an archival media and you erase the online backup directory contents.) If you have not archived the last backup you may not want the new scheduled backup to overwrite the old until you do so explicitly.

Example

The following example shows you how to use the Scheduling Manager to automatically run the following three commands:

- BACKUP SNAPSHOT.
- CHECK DATABASE.
- UPDATE STATISTICS.

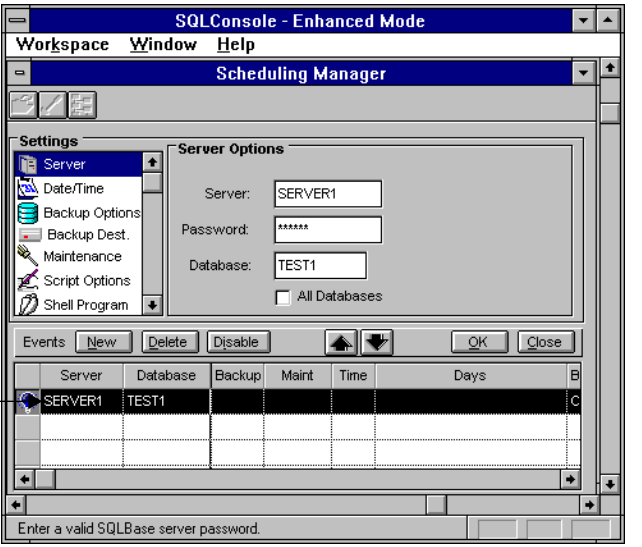
In this example we create three separate events, one for each command. This gives you more flexibility, if, for example, you want to perform CHECK DATABASE more frequently than you perform a backup. You can, though, create a single event that performs all three commands.

1. Double-click the Scheduling Manager node (under the Tools node on a SQLConsole workspace).

   The Scheduling Manager dialog box opens.

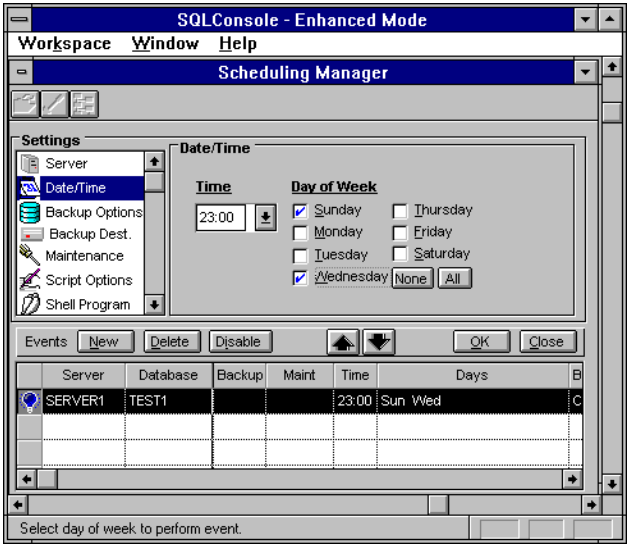
2. Click **New**.

3. Enter the server and database information.



SQLConsole automatically adds information about your event in this row.

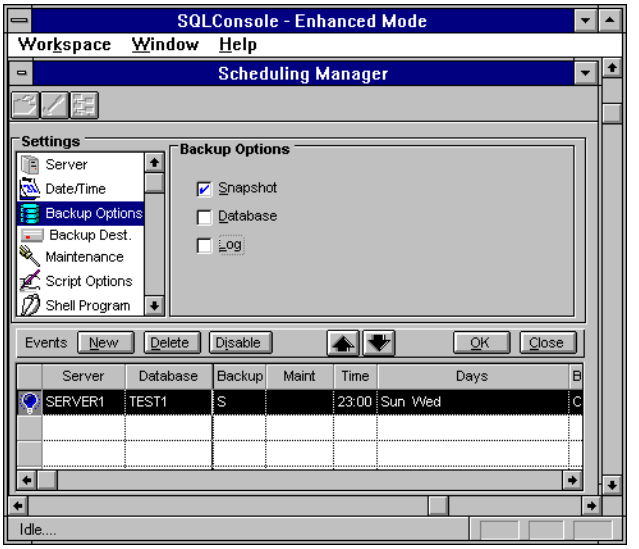
4. Click Date/Time in the Settings list box, then enter the date and time for the event.



Note: A single event can occur only at one particular time of the day (although you can schedule an event to occur on more than one day of the week). If you want an event to occur at two different times, you must schedule two separate events.

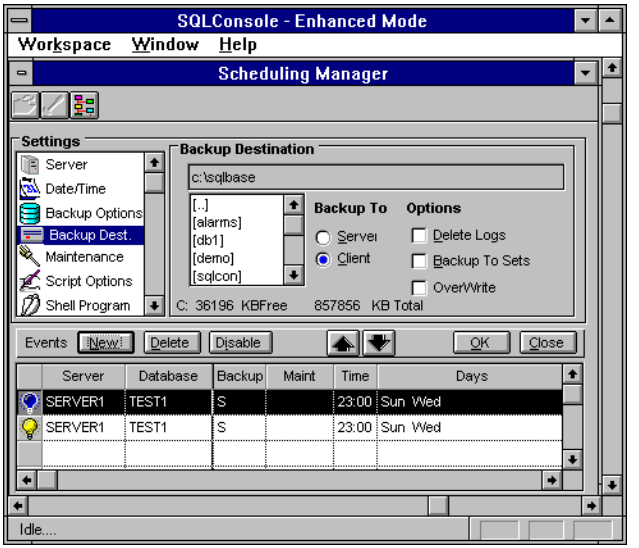
5. Click Backup Options (in the Settings list box) and then click Snapshot. In the row with information about your event, SQLConsole inserts an S in the Backup field.

For more information on backup options, refer to *Backing up a database* on page *6-19* and the *SQLBase Database Administrator's Guide*.



6. Click Backup Destination in the Settings list box. Accept the default information in the Backup Destination group box.

7. Click **New** to begin the next event. SQLConsole uses information from the previous event as default for the new event, so in the following steps you will customize the new event.



8. Click Date/Time in the Settings list box, then enter the date and time you want the event to occur.
9. Click Backup Options, then click to remove all checks from the group box.
10. Click Maintenance in the Settings list box, then click Check Database.
11. Click **New** to begin the next event.
12. Click Date/Time in the Settings list box, then enter the date and time you want the event to occur.
13. Click Backup Options, then click to remove all checks from the group box.
14. Click Maintenance in the Settings list box, then click Update Statistics.
15. Click **OK** to save the three events and close the window.

Order of execution

If you schedule two or more events at the same time, the event that occurs first in the Event list box runs first. You can change the relative order of the events in the list box with the **Up Arrow** and **Down Arrow** push buttons.

For events where more than one action is performed, the actions are performed in the following order:

1. Shell — Once regardless of databases selected.
2. Scripts — Once regardless of databases selected.
3. CHECK DATABASE — Once for each database.
4. CHECK TABLE — Only for specific database.
5. CHECK INDEX — Only for specific database.
6. UPDATE STATISTICS — Once for each database.
7. BACKUP SNAPSHOT or BACKUP DATABASE — Once for each database.
8. RELEASE LOG.
9. BACKUP LOGS.

Note: If partitions are enabled on a server, and you schedule a BACKUP SNAPSHOT or BACKUP DATABASE event, then the Scheduling Manager performs a BACKUP SNAPSHOT of the MAIN database.

Chapter 11

Security and Authorization

This chapter introduces the SQLBase security and authorization features available through SQLConsole:

- Database authority — This controls which users can log on to SQLBase and which actions they can perform.
- Table and view privileges — This describes how to share some data while keeping other data private.
- EXECUTE privileges — This controls which users can access stored procedures.

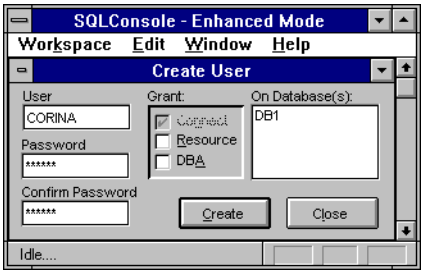
This information supplements information in the *SQLBase Database Administrator's Guide*. For more information, read the chapter on *Security and Authorization* in that manual.

Managing user accounts and information

You create and manage user accounts with the Users node (which appears under a database node) and its child nodes.

Create a user account

1. Right-click the Users node (under a database node) and then select **Create**.
2. Enter the appropriate information in the dialog box that appears.



3. Click **Create**.

   If you want to create additional users, repeat steps 2 and 3 as necessary.

4. Click **Close**.

Drop a user account

To drop a user account, right-click the node of that user (under the Users node, under a database node) and select **Drop**.

For example, to drop the Jo user account, right-click the Jo node (under the Users node) and select **Drop**.

Change a user's password

1. Right-click the user's node. For example, to change the password for user Jo, right-click the Jo node under the Users node.
2. Select **Alter**.
3. Enter the new password in the Password field and in the Confirm Password field, then close the window.

Change a user's authority

1. Right-click the user's node. For example, to change the authority for user Jo, right-click the Jo node under the Users node.
2. Select **Alter**.
3. Enter the new authority levels in the check boxes at the bottom of the window, then close the window.

   For more information on each authority level and revoking its authority, refer to the *SQLBase Database Administrator's Guide*.

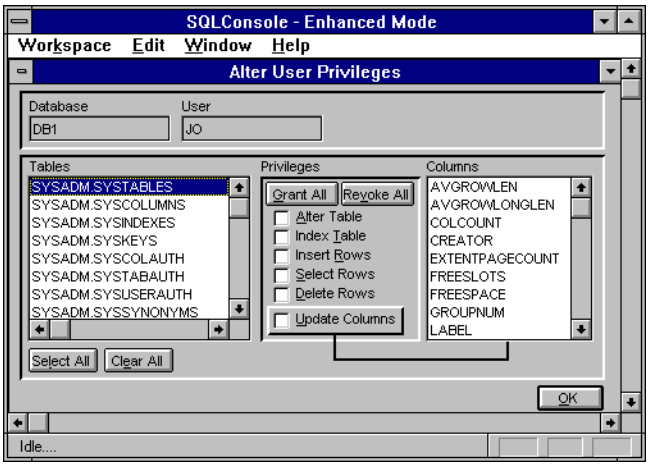
Table and view privileges

A user with DBA authority can grant privileges on *any* table or view in the database. A user with RESOURCE authority can grant privileges on only the tables that user created, or on the views based completely on tables the user created. Because a user with only CONNECT authority cannot create objects, this user cannot grant privileges, either.

A user who creates a table or view is the *owner* of that table or view and has all privileges on it. An owner can grant privileges to other users on the owner's objects.

Grant or change a user's table or view authority

1. Right-click the Table Authority (or View Authority) node under a particular user's node.
2. Select **Alter** and make the changes you want in the Alter User Privileges dialog box.



Grant or change privileges on a table or view

Use the following procedure to grant privileges on a particular table (or view) to one or more users. For more information on which privileges an owner can grant, refer to the *SQLBase Database Administrator's Guide.*

1. Find the table's node on a workspace, then right-click the Table Authority (or View Authority) node under it.
2. Select **Alter**, then make the changes you want in the Alter Table Privileges dialog box.

Synonyms

When you access a table or view created by another user (once you have been granted the privilege), you may want to fully-qualify the table name by prefixing it with the owner's name.

If you try to access another user's table or view in SQLTalk or with a SQL script and you do not qualify the table or view name with the owner's name, SQLBase looks for a table or view owned by *you* with that name.

Synonyms can save you typing by allowing you to refer to another user's table or view without having to fully qualify the name. A synonym is another name for a table or view.

For more information, read the *Synonyms* section in the *SQLBase Database Administrator's Guide*.

Create, alter, or drop a synonym

1. Right-click the Synonyms node (under a database node).
2. Select **Create**, **Alter**, or **Drop**. Note that all three options may not be enabled.
3. Enter the appropriate information in the dialog box that SQLConsole displays.

Views

Granting privileges is one way of giving users access to database objects. Creating views is another, more flexible way of giving users selective access to data. Views can enforce data security by limiting the rows or columns of data that users can access.

For more information on views, refer to the *SQLBase Database Administrator's Guide*.

To create or drop a view, right-click the Views node (under a database node), then select **Create** or **Drop**. Enter the appropriate information in the dialog box that appears.

EXECUTE privileges for stored procedures

To grant or change privileges for other users for stored procedures, right-click the Execute Authority node under a stored procedure's node.

You can either grant your own privileges to other users, or grant them privileges of their own.

Refer to the GRANT EXECUTE command section of the *SQLBase Database Administrator's Guide* for information.

Appendix A

Server and Database Properties

This appendix lists the server and database properties you can access and change through SQLConsole.

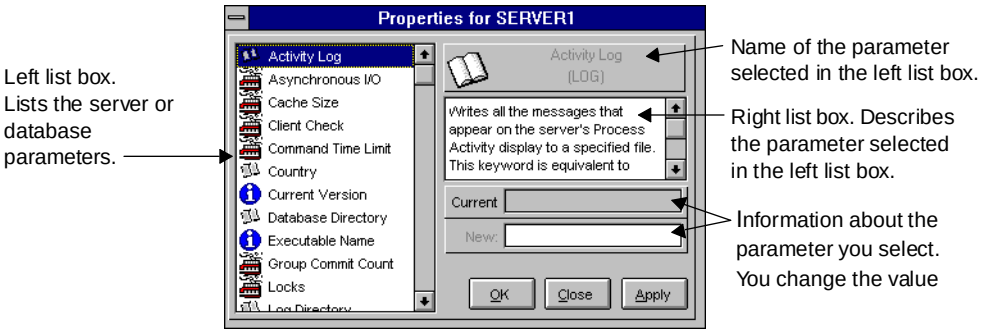
Database and server properties

You can easily view or change many server and database properties by using context menus.

View a server or database property

1. On a SQLConsole workspace, right-click a server node *or* a database node.
2. Select **Properties**.

   A server or database properties dialog box appears.

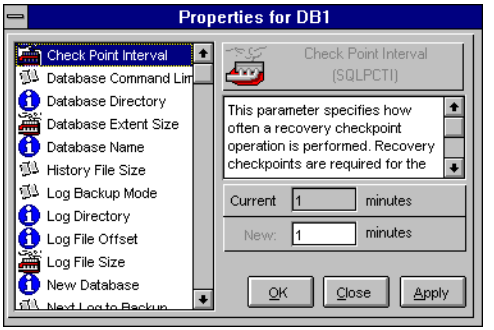


- In the left list box, items with the information icon (the lower case *i* in a blue circle) are *sqlget* parameters.
- Items with the book icon are general configuration settings and have no performance implications.
- Items with the engine icon are properties that can affect performance.

The *SQLBase SQL Application Programming Interface Reference* and the *SQLBase Database Administrator's Guide* provide more information about the properties.

Change a server or database property

1. Open a server or database properties dialog box as described in the previous procedure.



2. In the left list box, highlight the property you want to change, then make the change in the right-hand side of the dialog box.
3. Click **OK** to change the property and close the dialog box.

   OR

   Click **Apply** to change the property and leave the dialog box open (if you want to make more changes). When you are finished applying changes, click **Close**.

Some changes require that SQLBase be restarted to allow them to take effect. If this is the case, SQLConsole tells you.

Server properties

Using a server's context menu, you can view or change the following server properties. If the property is also a *sqlget* or *sqlset* parameter, the parameter is listed in uppercase following the parameter:

- Activity log (*log* keyword).
- Cache size (*cache* keyword).
- *Clientcheck* keyword.
- Command time limit (CMDTIMEOUT).
- *Country* keyword.
- Current version (SQLPVER).
- Database directory (*dbdir* keyword; SQLPDDR).
- Direct I/O (*directio* keyword).
- Executable name (SQLPEXE).

- Group commit count (*groupcommit* keyword).
- *Locks* keyword.
- Log directory (*logdir* keyword; SQLPLDR).
- *Netcheck* keyword.
- *Netchecktype* keyword.
- *Netlog* keyword.
- Number of deadlocks (SQLPDLK).
- Oracle outer join (*oracleouterjoin* keyword).
- Partitions (*partitions* keyword; SQLPPAR).
- Print level (SQLPPLV).
- Process timer (SQLPAPT).
- Silent mode *(silent* keyword).
- Server name *(servername* keyword).
- Statistics gathering (SQLPSTA).
- Sortcache size (*sortcache* keyword).
- Temporary file directory (*tempdir* keyword).
- Time colon only *(timecolononly* keyword).
- *Timeout* keyword.
- Timestamp (SQLPTMS).
- *Timezone* keyword.
- User limit (*users* keyword).
- Work space allocation (*workalloc* keyword).
- *Worklimit* keyword.

Database properties

Using a database's context menu, you can view or change the following database properties. If the property is also a *sqlget* or *sqlset* parameter, the parameter is listed in uppercase following the parameter:

- Checkpoint interval (SQLPTCI).
- Database command limit (SQLPDTL).
- Database directory (*dbdir* keyword; SQLPDDR).
- Database extent size (SQLPEXS).
- Database name (*dbname* keyword).
- History file size (SQLPHFS).

- Log backup mode (SQLPLBM).
- Log directory (*logdir* keyword; SQLPLDR).
- Log file offset (SQLPLGF).
- Log file size (SQLPLFS).
- New database (SQLPNDB).
- Next log to backup (SQLPNLB).
- Partitioned database (*partitions* keyword; SQLPPAR).
- Preallocate log files (SQLPPLF).
- Read-only database (SQLPROD).
- Read-only mode (*readonly* keyword).
- Recovery (SQLPREC).
- Transaction span limit (SQLPTSL).
- Catalog version counter (SQLPCVC).

Glossary

access path—The path used to get the data specified in a SQL command. An access path can involve an index or a sequential search (table scan), or a combination of the two. Alternate paths are judged based on the efficiency of locating the data.

aggregate function—A SQL operation that produces a summary value from a set of values.

alias—An alternative name used to identify a database object.

API (application programming interface)—A set of functions that a program uses to access a database.

application—A program written by or for a user that applies to the user's work. A program or set of programs that perform a task. For example, a payroll system.

argument—A value entered in a command that defines the data to operate on or that controls execution. Also called parameter or operand.

arithmetic expression—An expression that contains operations and arguments that can be reduced to a single numeric value.

arithmetic operator—A symbol used to represent an arithmetic operation, such as the plus sign (+) or the minus sign (-).

attribute—A characteristic or property. For example, the data type or length of a row. Sometimes, attribute is used as a synonym for column or field.

audit file—A log file that records output from an audit operation.

audit message—A message string that you can include in an audit file

audit operation—A SQLBase operation that logs database activities and performance, writing output to an audit file. For example, you can monitor who logs on to a database and what tables they access, or record command execution time.

authorization—The right granted to a user to access a database.

authorization-ID—A unique name that identifies a user. Associated to each authorization-id is a password. Abbreviated auth-id. Also called username.

back-end—See database server.

backup—To copy information onto a diskette, fixed disk, or tape for record keeping or recovery purposes.

base table—The permanent table on which a view is based. A base table is created with the CREATE TABLE command and does not depend on any other table. A base table has its description and its data physically stored in the database. Also called underlying table.

bindery—A NetWare 3.x database that contains information about network resources such as a SQLBase database server.

bind variable—A variable used to associate data to a SQL command. Bind variables can be used in the VALUES clause of an INSERT command, in a WHERE clause, or in the SET clause of an UPDATE command. Bind variables are the mechanism to transmit data between an application work area and SQLBase. Also called into variable or substitution variable.

browse—A mode where a user queries some of a database without necessarily making additions or changes. In a browsing application, a user needs to examine data before deciding what to do with it. A browsing application allows the user to scroll forward and backward through data.

buffer—A memory area used to hold data during input/output operations.

C/API—A language interface that lets a programmer develop a database application in the C programming language. The C/API has functions that a programmer calls to access a database using SQL commands.

cache—A temporary storage area in computer memory for database pages being accessed and changed by database users. A cache is used because it is faster to read and write to computer memory than to a disk file.

Cartesian product—In a join, all the possible combinations of the rows from each of the tables. The number of rows in the Cartesian product is equal to the number of rows in the first table times the number of rows in the second table, and so on. A Cartesian product is the first step in joining tables. Once the Cartesian product has been formed, the rows that do not satisfy the join conditions are eliminated.

cascade—A delete rule which specifies that changing a value in the parent table automatically affects any related rows in the dependent table.

case sensitive—A condition in which names must be entered in a specific lower-case, upper-case, or mixed-case format to be valid.

cast—The conversion between different data types that represent the same data.

CHAR—A column data type that stores character strings with a user-specified length. SQLBase stores CHAR columns as variable-length strings. Also called VARCHAR.

character—A letter, digit, or special character (such as a punctuation mark) that is used to represent data.

character string—A sequence of characters treated as a unit.

checkpoint—A point at which database changes older than the last checkpoint are flushed to disk. Checkpoints are needed to ensure crash recovery.

clause—A distinct part of a SQL command, such as the WHERE clause; usually followed by an argument.

client—A computer that accesses shared resources on other computers running as servers on the network. Also called front-end or requester.

column—A data value that describes one characteristic of an entity. The smallest unit of data that can be referred to in a row. A column contains one unit of data in a row of a table. A column has a name and a data type. Sometimes called field or attribute.

command—A user request to perform a task or operation. In SQLTalk, each command starts with a name, and has clauses and arguments that tailor the action that is performed. A command can include limits or specific terms for its execution, such as a query for names and addresses in a single zip code. Sometimes called statement.

commit—A process that causes data changed by an application to become part of the physical database. Locks are freed after a commit (except when cursor-context preservation is on). Before changes are stored, both the old and new data exist so that changes can be stored or the data can be restored to its prior state.

commit server—A database server participating in a distributed transaction, that has commit service enabled. It logs information about the distributed transaction and assists in recover after a network failure.

composite primary key—A primary key made up of more than one column in a table.

concatenated key—An index that is created on more than one column of a table. Can be used to guarantee that those columns are unique for every row in the table and to speed access to rows via those columns.

concatenation—Combining two or more character strings into a single string.

concurrency—The shared use of a database by multiple users or application programs at the same time. Multiple users can execute database transactions simultaneously without interfering with each other. The database software ensures that all users see correct data and that all changes are made in the proper order.

configure—To define the features and settings for a database server or its client applications.

connect—To provide a valid authorization-id and password to log on to a database.

connection handle—Used to create multiple, independent connections. An application must request a connection handle before it opens a cursor. Each connection handle represents a single transaction and can have multiple cursors. An application may request multiple connection handles if it is involved in a sequence of transactions.

consistency—A state that guarantees that all data encountered by a transaction does not change for the duration of a command. Consistency ensures that uncommitted updates are not seen by other users.

constant—Specifies an unchanging value. Also called literal.

control file—An ASCII file containing information to manage segmented load/unload files.

cooperative processing—Processing that is distributed between a client and a server in a such a way that each computer works on the parts of the application that it is best at handling.

coordinator—The application that initiates a distributed transaction.

correlated subquery—A subquery that is executed once for each row selected by the outer query. A subquery cannot be evaluated independently because it depends on the outer query for its results. Also called a repeating query. Also see subquery and outer query.

correlation name—A temporary name assigned to a table in an UPDATE, DELETE, or SELECT command. The correlation name and column name are combined to refer to a column from a specific table later in the same command. A correlation name is used when a reference to a column name could be ambiguous. Also called range variable.

crash recovery—The procedures that SQLBase uses automatically to bring a database to a consistent state after a failure.

current row—The latest row of the active result set which has been fetched by a cursor. Each subsequent fetch retrieves the next row of the active result set.

cursor—The term cursor refers to one of the following definitions:

- The position of a row within a result table. A cursor is used to retrieve rows from the result table. A named cursor can be used in the CURRENT OF clause or the ADJUSTING clause to make updates or deletions.
- A work space in memory that is used for gaining access to the database and processing a SQL command. This work space contains the return code, number of rows, error position, number of select list items, number of bind variables, rollback flag, and the command type of the current command.
- When the cursor belongs to an explicit connection handle that is created using the SQL/API function call *sqlcch* or the SQLTalk BEGIN CONNECTION command, it identifies a task or activity within a transaction. The task or activity can be compiled/executed independently within a single connection thread.

  Cursors can be associated with specific connection handles, allowing multiple transactions to the same database within a single application. When this is implemented, only one user is allowed per transaction.
- When a cursor belongs to an implicit connection handle created using the SQL/API function call *sqlcnc* or *sqlcnr*, or the SQLTalk CONNECT command, the cursor applies to an application in which you are connecting the cursor to a specific database that belongs to a single transaction.

cursor-context preservation—A feature of SQLBase where result sets are maintained after a COMMIT. A COMMIT does not destroy an active result set (cursor context). This enables an application to maintain its position after a COMMIT, INSERT, or UPDATE. For fetch operations, locks are kept on pages required to maintain the fetch position.

cursor handle—Identifies a task or activity within a transaction. When a connection handle is included in a function call to open a new cursor, the function call returns a cursor handle. The cursor handle can be used in subsequent SQL/API calls to identify the connection thread. A cursor handle is always part of a specific transaction and cannot be used in multiple transactions. However, a cursor handle can be associated with a specific connection handle. The ability to have multiple transactions to the same database within a single application is possible by associating cursor handles with connection handles.

Cursor Stability (CS)—The isolation level where a page acquires a shared lock on it only while it is being read (while the cursor is on it). A shared lock is dropped as the cursor leaves the page, but an exclusive lock (the type of lock used for an update) is retained until the transaction completes. This isolation level provides higher concurrency than Read Repeatability, but consistency is lower.

data dictionary—See system catalog.

data type—Any of the standard forms of data that SQLBase can store and manipulate. An attribute that specifies the representation for a column in a table. Examples of data types in SQLBase are CHAR (or VARCHAR), LONG VARCHAR (or LONG), NUMBER, DECIMAL (or DEC), INTEGER (or INT), SMALLINT, DOUBLE PRECISION, FLOAT, REAL, DATETIME (or TIMESTAMP), DATE, TIME.

database—A collection of interrelated or independent pieces of information stored together without unnecessary redundancy. A database can be accessed and operated upon by client applications such as SQLTalk.

database administrator (DBA)—A person responsible for the design, planning, installation, configuration, control, management, maintenance, and operation of a DBMS and its supporting network. A DBA ensures successful use of the DBMS by users.

A DBA is authorized to grant and revoke other users' access to a database, modify database options that affect all users, and perform other administrative functions.

database area—A database area corresponds to a file. These areas can be spread across multiple disk volumes to take advantage of parallel disk input/output operations.

database management system (DBMS)—A software system that manages the creation, organization, and modification of a database and access to data stored within it. A DBMS provides centralized control, data independence, and complex physical structures for efficient access, integrity, recovery, concurrency, and security.

database object—A table, view, index, synonym or other object created and manipulated through SQL.

database server—A DBMS that a user interacts with through a client application on the same or a different computer. Also called back-end.

DATE—A column data type in SQL that represents a date value as a three-part value (day, month, and year).

date/time value—A value of the data type DATE, TIME, or TIMESTAMP.

DCL (Data Control Language)—SQL commands that assign database access privileges and security such as GRANT and REVOKE.

DDL (Data Definition Language)—SQL commands that create and define database objects such as CREATE TABLE, ALTER TABLE, and DROP TABLE.

deadlock—A situation when two transactions, each having a lock on a database page, attempt to acquire a lock on the other's database page. One type of deadlock is where each transaction holds a shared lock on a page and each wishes to acquire an exclusive lock. Also called deadly embrace.

DECIMAL—A column data type that contains numeric data with a decimal point. Also called DEC.

default—An attribute, value, or setting that is assumed when none is explicitly specified.

delimited identifier—An identifier enclosed between two double quote characters (“) because it contains reserved words, spaces, or special characters.

delimiter—A character that groups or separates items in a command.

dependent object—An object whose existence depends on another object.

For example, if a stored procedure calls an external function, the stored procedure is the dependent object of the external function, since its existence depends on the external function.

dependent table—The table containing the foreign key.

determinant object—An object that determines the existence of another object.

For example, if a stored procedure calls an external function, the external function is the determinant object, since it determines the existence of the stored procedure.

dirty page—A database page in cache that has been changed but has not been written back to disk.

distributed database—A database whose objects reside on more than one system in a network of systems and whose objects can be accessed from any system in the network.

distributed transaction—Coordinates SQL statements among multiple databases that are connected by a network.

DLL (Dynamic Link Library)—A program library written in C or assembler that contains related modules of compiled code. The functions in a DLL are not read until run-time (dynamic linking).

DML (Data Manipulation Language)—SQL commands that change data such as INSERT, DELETE, UPDATE, COMMIT, and ROLLBACK.

DOUBLE PRECISION—A column data type that stores a floating point number.

DQL (Data Query Language)—The SQL SELECT command, which lets a user request information from a database.

duplicates—An option used when creating an index for a table that specifies whether duplicate values are allowed for a key.

embedded SQL—SQL commands that are embedded within a program, and are prepared during precompilation and compilation before the program is executed. After a SQL command is prepared, the command itself does not change (although values of host variables specified within the command can change). Also called static SQL.

entity—A person, place, or thing represented by a table. In a table, each row represents an entity.

equijoin—A join where columns are compared on the basis of equality, and all the columns in the tables being joined are included in the results.

Ethernet—A LAN with a bus topology (a single cable not connected at the ends). When a computer wants to transmit, it first checks to see if another computer is transmitting. After a computer transmits, it can detect if a collision has happened. Ethernet is a broadcast network and all computers on the network hear all transmissions. A computer selects only those transmissions addressed to it.

exclusive lock (X-lock)—An exclusive lock allows only one user to have a lock on a page at a time. An exclusive lock prevents another user from acquiring a lock until the exclusive lock is released. Exclusive locks are placed when a page is to be modified (such as for an UPDATE, INSERT, or DELETE).

An exclusive lock differs from a shared lock because it does not permit another user to place any type of lock on the same data.

expression—An item or a combination of items and operators that yield a single value. Examples are column names which yield the value of the column in successive rows, arithmetic expressions built with operators such as + or - that yield the result of performing the operation, and functions which yield the value of the function for its argument.

extent page—A database page used when a row is INSERTed that is longer than a page or when a row is UPDATEd and there is not enough space in the original page to hold the data.

external function—A user-defined function that resides in an "external" DLL (Dynamic Link Library) invoked within a SQLBase stored procedure.

event—See timer event.

field—See column.

file server—A computer that allows network users to store and share information.

FLOAT—A column data type that stores floating point numbers.

floating point—A number represented as a number followed by an exponent designator (such as 1.234E2, -5.678E2, or 1.234E-2). Also called E-notation or scientific notation.

foreign key—Foreign keys logically connect different tables. A foreign key is a column or combination of columns in one table whose values match a primary key in another table. A foreign key can also be used to match a primary key within the same table.

front-end—See client.

function—A predefined operation that returns a single value per row in the output result table.

grant—That act of a system administrator to permit a user to make specified use of a database. A user may be granted access to an entire database or specific portions, and have unlimited or strictly-limited power to display, change, add, or delete data.

GUI (Graphical User Interface)—A graphics-based user interface with windows, icons, pull-down menus, a pointer, and a mouse. Microsoft Windows and Presentation Manager are examples of graphical user interfaces.

history file—Contains previous versions of changed database pages. Used when read-only (RO) isolation level is enabled.

host language—A program written in a language that contains SQL commands.

identifier—The name of a database object.

index—A data structure associated with a table used to locate a row without scanning an entire table. An index has an entry for each value found in a table's indexed column or columns, and pointers to rows having that value. An index is logically ordered by the values of a key. Indexes can also enforce uniqueness on the rows in a table.

INTEGER—A column data type that stores a number without a decimal point. Also call INT.

isolation level—The extent to which operations performed by one user can be affected by (are isolated from) operations performed by another user. The isolation levels are Read Repeatability (RR), Cursor Stability (CS), Release Locks (RL), and Read Only (RO).

join—A query that retrieves data from two or more tables. Rows are selected when columns from one table match columns from another table. See also Cartesian product, self-join, equijoin, natural join, theta join, and outer join.

key—A column or a set of columns in an index used to identify a row. A key value can be used to locate a row.

keyword—One of the predefined words in a command language.

local area network (LAN)—A collection of connected computers that share data and resources, and access other networks or remote hosts. Usually, a LAN is geographically confined and microcomputer-based.

lock—To temporarily restrict other users access to data to maintain consistency. Locking prevents data from being modified by more than one user at a time and prevents data from being read while being updated. A lock serializes access to data and prevents simultaneous updates that might result in inconsistent data. See shared lock (S-lock) and exclusive lock (X-lock).

logical operator—A symbol for a logical operation that connects expressions in a WHERE or HAVING clause. Examples are AND, OR, and NOT. An expression formed with logical operators evaluates to either TRUE or FALSE. Logical operators define or limit the information sought. Also called Boolean operator.

LONG VARCHAR—In SQL, a column data type where the value can be longer than 254 bytes. The user does not specify a length. SQLBase stores LONG VARCHAR columns as variable-length strings. Also called LONG.

mathematical function—An operation such as finding the average, minimum, or maximum value of a set of values.

media recovery—Restoring data from backup after events such as a disk head crash, operating system crash, or a user accidentally dropping a database object.

message buffer—The input message buffer is allocated on both the client computer and the database server. The database server builds an input message in this buffer on the database server and sends it across the network to a buffer on the client. It is called an input message buffer because it is input from the client's point of view.

The out put message buffer is allocated on both the client computer and on the database server. The client builds an output message in this buffer and sends it to a buffer on the database server. It is called an output message buffer because it is output from the client's point of view.

modulo—An arithmetic operator that returns an integer remainder after a division operation on two integers.

multi-user—The ability of a computer system to provide its services to more than one user at a time.

natural join—An equijoin where the value of the columns being joined are compared on the basis of equality. All the columns in the tables are included in the results but only one of each pair of joined columns is included.

NDS (NetWare Directory Services)—A network-wide directory included with NetWare 4.x, that provides global access to all network resources, regardless of their physical location. The directory is accessible from multiple points by network users, services and applications.

nested query—See subquery.

NetWare—The networking components sold by Novell. NetWare is a collection of data link drivers, a transport protocol stack, client computer software, and the NetWare server operating system. NetWare runs on Token Ring, Ethernet, and ARCNET.

NetWare 386—A server operating system from Novell for computers that controls system resources on a network.

NLM (NetWare Loadable Module)—An NLM is a NetWare program that you can load into or unload from server memory while the server is running. When loaded, an NLM is part of the NetWare operating system. When unloaded, an NLM releases the memory and resources that were allocated for it.

null—A value that indicates the absence of data. Null is not considered equivalent to zero or to blank. A value of null is not considered to be greater than, less than, or equivalent to any other value, including another value of null.

NUMBER—A column data type that contains a number, with or without a decimal point and a sign.

numeric constant—A fixed value that is a number.

ODBC—The Microsoft Open DataBase Connectivity (ODBC) standard, which is an application programming interface (API) specification written by Microsoft. It calls for all client applications to write to the ODBC standard API and for all database vendors to provide support for it. It then relies on third-party database drivers or access tools that conform to the ODBC specification to translate the ODBC standard API calls generated by the client application into the database vendor's proprietary API calls.

operator—A symbol or word that represents an operation to be performed on the values on either side of it. Examples of operators are: arithmetic (+, -, *, /), relational (=, !=, >, <, >=, <=), and logical (AND, OR, NOT).

optimization—The determination of the most efficient access strategy for satisfying a database access.

outer join—A join in which both matching and non-matching rows are returned. Each preserved row is joined to an imaginary row in the other table in which all the fields are null.

outer query—When a query is nested within another query, the main query is called the outer query and the inner query is called the subquery. An outer query is executed once for each row selected by the subquery. A subquery cannot be evaluated independently but that depends on the outer query for its results. Also see subquery.

page—The physical unit of disk storage that SQLBase uses to allocate space to tables and indexes.

parent table—The table containing the primary key.

parse—To examine a command to make sure that it is properly formed and that all necessary information is supplied.

partitioning—A method of setting up separate user areas to maximize disk space. Databases can be stretched across several different network partitions.

password—A sequence of characters that must be entered to connect to a database. Associated to each password is an authorization-id.

picture—A string of characters used to format data for display.

precedence—The default order in which operations are performed in an expression.

precision—The maximum number of digits in a column.

precompilation—Processing of a program containing SQL commands or procedures that takes place before compilation. SQL commands are replaced with statements that are recognized by the host language compiler. Output from precompilation includes source code that can be submitted to the compiler.

predicate—An element in a search condition that expresses a comparison operation that states a set of criteria for the data to be returned by a query.

primary key—The columns or set of columns that are used to uniquely identify each row in a table. All values for a key are unique and non-null.

privilege—A capability given to a user to perform an action.

procedure—A named set of SAL or SQL statements that can contain flow control language. You compile a procedure for immediate and/or later execution.

query—A request for information from a database, optionally based on specific conditions. For example, a request to list all customers whose balance is greater than $1000. Queries are issued with the SELECT command.

Read Only (RO)—The isolation level where pages are not locked, and no user has to wait. This gives the user a snapshot view of the database at the instant that the transaction began. Data cannot be updated while in the read-only isolation level.

Read Repeatability (RR)—The isolation level where if data is read again during a transaction, it is guaranteed that those rows would not have changed. Rows referenced by the program cannot be changed by other programs until the program reaches a commit point. Subsequent queries return a consistent set of results (as though changes to the data were suspended until all the queries finished). Other users will not be able to update any pages that have been read by the transaction. All shared locks and all exclusive locks are retained on a page until the transaction completes. Read repeatability provides maximum protection from other active application programs. This ensures a high level of consistency, but lowers concurrency. SQLBase default isolation level.

REAL—A column data type that stores a single-precision number.

record—See row.

recovery—Rebuilding a database after a system failure.

referential cycle—Tables which are dependents of one another.

referential integrity—Guarantees that all references from one database table to another are valid and accurate. Referential integrity prevents problems that occur because of changes in one table which are not reflected in another.

relation—See table.

relational database—A database that is organized and accessed according to relationships between data items. A relational database is perceived by users as a collection of tables.

relational operator—A symbol (such as =, >, or <) used to compare two values. Also called comparison operator.

Release Locks (RL)—With the Cursor Stability isolation level, when a reader moves off a database page, the shared lock is dropped. However, if a row from the page is still in the message buffer, the page is still locked.

In contrast, the Release Lock (RL) isolation level increases concurrency. By the time control returns to the application, all shared locks have been released.

repeating query—See correlated subquery.

requester—See client.

restore—Copying a backup of a database or its log files to a database directory.

restriction mode—In restriction mode, the result set of one query is the basis for the next query. Each query further restricts the result set. This continues for each subsequent query.

result set mode—Normally, result table rows are displayed and scrolled off the screen. In result set mode, the rows of the result table are available for subsequent scrolling and retrieval.

result table—The set of rows retrieved from one or more tables or views during a query. A cursor allows the rows to be retrieved one by one.

revoke—The act of withdrawing a user's permission to access a database.

rollback—To restore a database to the condition it was in at its last COMMIT. A ROLLBACK cancels a transaction and undoes any changes that it made to the database. All locks are freed unless cursor-context preservation is on.

rollforward—Reapplying changes to a database. The transaction log contains the entries used for rollforward.

router—A client application talks to a SQLBase server through a router program. The router enables a logical connection between a client and the server. Once this connection is established on the LAN, the client application uses the router program to send SQL requests to the server and to receive the results.

row—A set of related columns that describe a specific entity. For example, a row could contain a name, address, telephone number. Sometimes called record or tuple.

ROWID—A hidden column associated with each row in a SQLBase table that is an internal identifier for the row. The ROWID can be retrieved like any other column.

ROWID validation—A programming technique that ensures that a given row that was SELECTed has not been changed or deleted by another user during a session. When a row is updated, the ROWID is changed.

SAP (Service Advertisement Protocol)—A NetWare protocol that resources (such as database servers) use to publicize their services and addresses on a network.

savepoint—An intermediate point within a transaction to which a user can later ROLLBACK to cancel any subsequent commands, or COMMIT to complete the commands.

scale—The number of digits to the right of the decimal point in a number.

search condition—A criterion for selecting rows from a table. A search condition appears in a WHERE clause and contains one or more predicates.

search—To scan one or more columns in a row to find rows that have a certain property.

self-join—A join of a table with itself. The user assigns the two different correlation names to the table that are used to qualify the column names in the rest of the query.

self-referencing table—A table that has foreign and primary keys with matching values within the same table.

server—A computer on a network that provides services and facilities to client applications.

shared lock (S-lock)—A shared lock permits other users to read data, but not to change it. A shared lock lets users read data concurrently, but does not let a user acquire an exclusive lock on the data until all the users' shared locks have been released. A shared lock is placed on a page when the page is read (during a SELECT). At a given time, more than one user can have a shared lock placed on a page. The timing of the release of a shared lock depends on the isolation level.

A shared lock differs from an exclusive lock because it permits more than one user to place a lock on the same data.

single-user—A computer system that can only provide its services to one user at a time.

SMALLINT—A column data type that stores numbers without decimal points.

socket—An identifier that Novell's IPX (Internetwork Packet Exchange) uses to route packets to a specific program.

SPX (Sequenced Packet Exchange)—A Novell communication protocol that monitors network transmissions to ensure successful delivery. SPX runs on top of Novell's IPX (Internetwork Packet Exchange).

SQL (Structured Query Language)—A standard set of commands used to manage information stored in a database. These commands let users retrieve, add, update, or delete data. There are four types of SQL commands: Data Definition Language (DDL), Data Manipulation Language (DML), Data Query Language (DQL), and Data Control Language (DCL). SQL commands can be used interactively or they can be embedded within an application program. Pronounced ess-que-ell or sequel.

SQLBase—A relational DBMS that lets users access, create, and update data.

SQLTalk—SQLTalk is an interactive user interface for SQLBase that is used to manage a relational database. SQLTalk has a complete implementation of SQL and many extensions. SQLTalk is a client application.

static SQL—See embedded SQL.

statistics—Attributes about tables such as the number of rows or the number of pages. Statistics are used during optimization to determine the access path to a table.

storage group—A list of database areas. Storage groups provide a means to allow databases or tables to be stored on different volumes.

stored procedure—A precompiled procedure that is stored on the backend for future execution.

string delimiter—A symbol used to enclose a string constant. The symbol is the single quote (').

string—A sequence of characters treated as a unit of data.

subquery—A SELECT command nested within the WHERE or HAVING clause of another SQL command. A subquery can be used anywhere an expression is allowed if the subquery returns a single value. Sometimes called a nested query. Also called subselect. See also correlated subquery.

synonym—A name assigned to a table, view, external function that may be then used to refer to it. If you have access to another user's table, you may create a synonym for it and refer to it by the synonym alone without entering the user's name as a qualifier.

syntax—The rules governing the structure of a command.

system catalog—A set of tables SQLBase uses to store metadata. System catalog tables contain information about database objects, privileges, events, and users. Also called data dictionary.

system keywords—Keywords that can be used to retrieve system information in commands.

table—The basic data storage structure in a relational database. A table is a two-dimensional arrangement of columns and rows. Each row contains the same set of data items (columns). Sometimes called a relation.

table scan—A method of data retrieval where a DBMS directly searches all rows in a table sequentially instead of using an index.

theta join—A join that uses relational operators to specify the join condition.

TIME—A column data type in the form of a value that designates a time of day in hours, minutes, and possibly seconds (a two- or three-part value).

timeout—A time interval allotted for an operation to occur.

TIMESTAMP—A column data type with a seven-part value that designates a date and time. The seven parts are year, month, day, hour, minutes, seconds, and microseconds (optional). The format is:

```
yyyy-mm-dd-hh.mm.ss.nnnnnn
```

timer event—Executes a procedure at a predetermined time. You can optionally repeat the timer event at specified intervals.

token—A character string in a specific format that has some defined significance in a SQL command.

Token-Ring—A LAN with ring topology (cable connected at the ends). A special data packet called a token is passed from one computer to another. When a computer gets the token, it can attach data to it and transmit. Each computer passes on the data until it arrives at its destination. The receiver marks the message as being received and sends the message on to the next computer. The message continues around the ring until the sender receives it and frees the token.

tokenized error message—An error message formatted with tokens in order to provide users with more informational error messages. A tokenized error message contains one or more variables that SQLBase substitutes with object names (tokens) when it returns the error message to the user.

transaction—A logically-related sequence of SQL commands that accomplishes a particular result for an application. SQLBase ensures the consistency of data by verifying that either all the data changes made during a transaction are performed, or that none of them are performed. A transaction begins when the application starts or when a COMMIT or ROLLBACK is executed. The transaction ends when the next COMMIT or ROLLBACK is executed. Also called logical unit of work.

transaction log—A collection of information describing the sequence of events that occur while running SQLBase. The information is used for recovery if there is a system failure. A log includes records of changes made to a database. A transaction log in SQLBase contains the data needed to perform rollbacks, crash recovery, and media recovery.

trigger—Activates a stored procedure that SQLBase automatically executes when a user attempts to change the data in a table, such as on a DELETE or UPDATE command.

two-phase commit—The protocol that coordinates a distributed transaction commit process on all participating databases.

tuple—See row.

unique key—One or more columns that must be unique for each row of the table. An index that ensures that no identical key values are stored in a table.

username—See authorization-id.

value—Data assigned to a column, a constant, a variable, or an argument.

VARCHAR—See CHAR.

variable—A data item that can assume any of a given set of values.

view—A logical representation of data from one or more base tables. A view can include some or all of the columns in the table or tables on which it is defined. A view represents a portion of data generated by a query. A view is derived from a base table or base tables but has no storage of its own. Data for a view can be updated in the same manner as for a base table. Sometimes called a virtual table.

wildcard—Characters used in the LIKE predicate that can stand for any one character (the underscore _) or any number of characters (the percent sign %) in pattern-matching.

Windows—A graphical user interface from Microsoft that runs under DOS.

With Windows, commands are organized in lists called menus. Icons (small pictures) on the screen represent applications. A user selects a menu item or an icon by pointing to it with a mouse and clicking.

Applications run in windows that can be resized and relocated. A user can run two or more applications at the same time and can switch between them. A user can run multiple copies of the same application at the same time.

write-ahead log (WAL)—A transaction logging technique where transactions are recorded in a disk-based log before they are recorded in the physical database. This ensures that active transactions can be rolled back if there is a system crash.

Index

D

E

F

Q

R

S

T

U

V

W