About This Manual

This manual contains information about Pervasive.SQL version 7 utilities for both server and workstation products. These utilities allow you to perform tasks such as configuring server and workstation components, maintaining data files, and creating and updating DDFs.

Who Should Read This Manual

This manual provides information for users who install and run Pervasive.SQL 7 server and workstation products. This manual is also useful for system administrators who are responsible for maintaining databases on a network and for programmers who are using Pervasive.SQL to develop applications.

Pervasive Software would appreciate your comments and suggestions about this manual. As a user of our documentation, you are in a unique position to provide ideas that can have a direct impact on future releases of this and other manuals. Please complete the User Comments form that appears on our Web site and fill in part number 100-003441-003.

Manual Organization

- Chapter 1—“Introduction to Pervasive.SQL Utilities”

  This chapter provides a summary of the Pervasive.SQL utilities.

- Chapter 2—“Component Architecture”

  This chapter discusses smart component architecture, a new feature in Pervasive.SQL.

- Chapter 3—“Configuring Components Using the Setup Utility”

  This chapter describes the Configuration utility, which allows you to configure all Pervasive components.

- Chapter 4—“Creating and Maintaining DDFs with DDF Ease”

  This chapter provides detailed information about creating and maintaining DDFs for existing Btrieve data files.

- Chapter 5—“Monitoring Pervasive.SQL Database Resources”

  This chapter describes the Monitor utility, which monitors activities of the Btrieve and SQL engines.

- Chapter 6—“Testing Btrieve Operations Using the Function Executor”

  This chapter describes the Function Executor, which allows you to execute Btrieve operations one at a time.

- Chapter 7—“Manipulating Btrieve Data Files with the Maintenance Utility”

  This chapter describes the interactive and command-line Btrieve Maintenance utilities. These utilities allow you to perform common file and data manipulations on MicroKernel data files.

- Chapter 8—“Manipulating Scalable SQL Data Files with the Maintenance Utility”

  This chapter describes the SQL Interface Maintenance utility (SQLUTIL), a command-line utility that allows you to perform common file and data manipulation on relational data files.

- Chapter 9—“Executing SQL Statements with SQLScope”

  This chapter describes SQLScope, a Win16 interactive SQL script file editor and handler. SQLScope allows you to query data files using SQL statements and display the results on the screen. In addition, SQLScope allows you to perform database management tasks, such as importing and exporting SQL scripts.

- Chapter 10—“Checking and Repairing Referential Integrity”

  This chapter describes the RI utility, which allows you to check the consistency of data files containing referential constraints.

- Chapter 11—“Converting MicroKernel Data Files”

  This chapter describes how to rebuild previous versions of MicroKernel files into version 7.0 format.

- Appendix A—“Smart Component Type Codes”

  This appendix lists each smart component and type by group in tabular format.

- Appendix B—“Description Files”

  This appendix documents description files, which are used with the Btrieve Maintenance utilities. The appendix explains the rules for creating description files, provides description file examples, and describes the individual description file elements.

The manual also includes a glossary and index.

Conventions

Unless otherwise noted, command syntax, code, and examples use the following conventions:

Case
Commands and reserved words typically appear in uppercase letters. Unless the manual states otherwise, you can enter these items using uppercase, lowercase, or both. For example, you can type MYPROG, myprog, or MYprog.

[ ]
Square brackets enclose optional information, as in [*log_name*]. If information is not enclosed in square brackets, it is required.

|
A vertical bar indicates a choice of information to enter, as in [*file name* | *@file name*].

< >
Angle brackets enclose multiple choices for a required item, as in /D=<5|6|7> .

variable
Words appearing in italics are variables that you must replace with appropriate values, as in *file name* .

...
An ellipsis following information indicates you can repeat the information more than one time, as in [*parameter* ...].

::=
The symbol ::= means one item is defined in terms of another. For example, a::=b means the item *a* is defined in terms of *b* .

Introduction to Pervasive.SQL Utilities

Pervasive.SQL is a comprehensive database management system built around Pervasive Software's MicroKernel Database Engine. Pervasive.SQL 7 offers easy installation, uncomplicated maintenance, and high levels of performance and reliability. This manual describes the Pervasive.SQL 7 utilities you can use to configure and manage your Pervasive.SQL database.

- "Utility Summary"
- "File System Security"

Utility Summary

The following table summarizes these utilities.

Table 1-1
Summary of Pervasive.SQL 7 Utilities

Utility	Platforms	Description	Server or Workstation
Setup	Win16, Win32, and OS/2	Manipulates settings for Pervasive client and server components.	Windows NT and NetWare servers Windows 9*X* /NT workstations for Win16 and Win32 versions
Monitor	Win16, Win32, and OS/2	Monitors server engine activity, useful for database administration and programming diagnostics.	Windows NT servers and Netware servers from any platform other than NLM
Function Executor	Win16 and OS/2	Executes Btrieve operations, enabling you to learn how Btrieve works or test and debug an application.	Windows NT and NetWare servers Windows 9*X* /NT workstations for Win16 version
Btrieve Maintenance	DOS, NLM, Win16, Win32, and OS/2	Performs common Btrieve file and data manipulations, such as importing and exporting data.	Windows NT and NetWare servers Windows 9*X* /NT workstations for command-line and Win32 versions
Scalable SQL Maintenance	Win32 and NLM	Performs common SQL Interface file and data manipulations, such as importing and exporting data.	Windows NT and NetWare servers Windows 9*X* /NT workstations for command-line version
SQLScope	Win16 and Win32	Allows you to execute SQL Statements interactively.	Windows NT and NetWare servers Windows 9*X* /NT workstations
RI utility	NLM only	Checks and lists RI constraints on named databases.	NetWare Server only
Rebuild	Win16, Win32, OS/2, and NLM	Converts previous versions of MicroKernel files into version 7.0 format.	Windows NT and NetWare servers Windows 9*X* /NT workstations for Win32 version
View Conversion	Win32, DOS	Migrates Scalable SQL 3.*x* view to Scalable SQL 4.*x* .	Windows NT servers Windows 9*X* /NT workstations for Win32 version

DDF Ease	Win32	Creates and maintains Data Dictionary Files (DDFs) and database files.	Windows NT and NetWare servers Windows 9*X* /NT workstations
User Count Administrator	Win16 and Win32	Increases the Pervasive.SQL user count incrementally with a software key you obtain from Pervasive Software. This utility is documented in *Getting Started with* Pervasive.SQL (Server Edition).	Windows NT and NetWare servers
InstallScout	Win16 and Win32	To ensure that your system meets network communication requirements before installation and that your new software is performing correctly after installation. For more information, see the InstallScout Help file (INSSCT.HLP).	Windows NT and NetWare servers Windows 9*X* /NT workstations
SmartScout	Win16 and Win32	A troubleshooting utility that analyzes components, runs system tests, and allows you to display registry and .INI file settings. This utility is documented in *Getting* Started with Pervasive.SQL.	Windows NT and NetWare servers Windows 9*X* /NT workstations

File System Security

Pervasive.SQL engines adhere to the file system security defined by the specific operating system, such as NTFS and Novell's NSS.

Component Architecture

This chapter discusses the following topics:

Smart Components

Pervasive.SQL 7 offers a new component architecture called Smart Components, which improves installation and run-time reliability and makes application troubleshooting easier.

In earlier Pervasive software releases, some developers experienced one or more of the following problems:

- Installation of a new application (with old client Requester components) overwrote new Requester components in shared locations, causing old applications to fail the next time they ran.
- Installation of a new workstation engine was incompatible with existing client Requesters. Existing client Requesters loaded the old engine, failing to provide features required by the new application.
- Difficulty identifying the function, version, and patch level of installed components.
- Difficulty determining the root cause of run-time operational failures, especially in client/server operation.

The Smart Components architecture is designed to reduce or eliminate these problems by providing the following features and benefits:

- Component Identification. Component function, major, and minor functional level are easily identified to aid in problem resolution.
- Unique Component Naming. Each release of a given component has a unique file name, so that updated versions of a component never overwrite previous versions. A Pervasive upgrade will not damage existing Pervasive-based applications.
- Dynamic Binding. Pervasive.SQL no longer loads a fixed set of program files into memory. Dependent components are loaded only if another component specifically requires its functionality, major, and minor functional level. Incompatible components are never accidentally loaded, reducing or eliminating version-related failures.
- Pervasive.SQL Event Logging. All components report errors and messages to a central log, easing the burden of troubleshooting.
- Error Code Clarification. Error conditions from underlying layers are now logged through to the Pervasive.SQL Event Log, rather than hidden within an umbrella status code. Because the root causes of certain errors can now be more quickly determined, troubleshooting is much easier.

Component Identification

Each component contains a unique embedded Component ID. The Component ID is a string containing information such as:

- Designated operating system
- Functionality
- Major functional level
- Minor functional level
- Build site
- Build number
- Timestamp
- Checksum

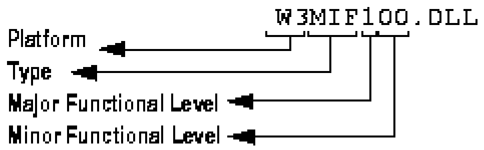
Pervasive Software Customer Support representatives can browse the file image of a component to locate the Component ID and verify that it is the correct component.

Unique Component Naming

Pervasive.SQL components have new unique names that reflect the platform, type, and functional level of the component. Each subsequent release of a component (even patches) will have a slightly different name, so that no two releases of the same functional component have the same file name.

This feature both identifies the exact functionality of a file and prevents different versions of a file from overwriting each other during installation of a new version or uninstall to a previous version.

Components are named using a well-defined scheme. All component names adhere to 8.3 notation for compatibility on systems that do not support long file names (such as Windows 3.1). The first two characters of the prefix identify the designated run-time platform. The next three characters identify the component functionality—its type. The sixth character identifies the major functional level (hexadecimal, range 1 to F), and the final two characters identify the minor functional level (hexadecimal, range 00 to FF).



The major functional level defines the version of the component, which began at one (1) with the first release of Pervasive.SQL. The minor functional level begins at zero (0) for each major functional level. Whenever the major functional level is incremented, the minor functional level is reset to zero. The minor functional level increases with each patch or public release of the component if it contains any changes whatsoever. The following tables show the Platform Codes and a sampling of Component Type Codes.

Table 2-1
Platform Codes

Platform	Code
Windows 3.1, Windows for Workgroups (Win16)	W1
Extended Windows (32-bit Watcom Extender)	W2
Windows 95, Windows NT (Win32)	W3
Windows 9X	W9
Windows NT	WT
NetWare 3.x and 4.x	NW
OS/2 (32-bit)	O3

Table 2-2
Component Type Codes**1

Component	Type Code
Btrieve Interface DLL	BIF
Network Services Layer	NSL
MicroKernel Interface DLL	MIF
SQL Interface DLL	SIF

***1* This table lists a sampling of Component Types. For a complete listing, refer to Appendix A, “Smart Component Type Codes.”

Dynamic Binding

Rather than load a hard-coded file name, upstream components that require the functionality of a downstream component now can specify what functionality and what revision they require. An upstream component loads before a downstream component. For example, the application loads the Btrieve interface component, or .BIF. The .BIF then loads the .MIF, or the MicroKernel interface module. In this example, the .BIF is an upstream component, and the .MIF is a downstream component.

Upstream components load their downstream components with the help of a new component, the Abstract OS Services DLL (Services DLL). The upstream component provides the Services DLL with a *binding rule* , specifying the type and minimum functional level of the required downstream component. Based on the binding rule, the Services DLL constructs a file name template and searches for a file that can deliver at least the required functionality. When it finds such a file, it returns the full path to the calling component, which then loads the specified downstream component.

When searching for a downstream component, the Services DLL first explores the directory or directories specified in the PERVASIVE_PATH environment variable. If PERVASIVE_PATH is not set or the requested component is not found, the Services DLL then searches on the platform's default path. (For OS/2, the Services DLL does not search LIBPATH.)

Note: The Services DLL does not search for the Glue DLLs and the initial Services DLL. Instead, the operating system uses the platform's default path.

The Services DLL employs a best first match search algorithm, meaning that it stops in the first directory where it finds an acceptable match and then returns the highest minor functional level of the specified component that exists in that directory.

For example, if a component requires W1MIF101.DLL or later, the Services DLL searches until it finds an instance of W1MIF1*xx* .DLL, where *xx* is 01 or greater. Then, the Services DLL searches that directory for the instance of W1MIF1*xx* .DLL with the greatest value of *xx* . This file name is then returned to the calling component. For example, if W1MIF101.DLL and W1MIF102.DLL are present in that directory, W1MIF102.DLL is returned. If there is an instance of W1MIF103.DLL in a different directory later on the search path, it is never reached.

Pervasive.SQL Event Logging

With the release of Pervasive.SQL 7, the MicroKernel message log is replaced by a new centralized event log. All Pervasive.SQL 7 and later components write status and error messages to the same log file. In addition, if two or more Pervasive-based applications are running on the same machine, they share a single event log.

The event log, called PVSW.LOG, is located in the Windows root directory of each machine that is running a Pervasive-based application. This location cannot be changed or customized. In the following table, C: represents the drive letter where your operating system is installed.

Platform	Event log location
Windows 9X and Windows 3.1x	C:\WINDOWS
Windows NT	C:\WINNT
NetWare	SYS:SYSTEM
OS/2	C:\OS2

Syntax

The event log consists of ASCII text messages that adhere to the following syntax description:

Table 2-3
Event Log Fields

Field	Length (in Bytes)	Contents
Date	10	Automatic date-stamp in *mm/dd/yyyy* format.
Time	8	Automatic time-stamp in *hh:mm:ss* format.
Component	15	File name of component returning the error (prefix only, no extension).
Process	8	Instance ID of the component, which is either the process ID of the component or the thread group ID in NetWare.
Process Name	Up to 15	Path and name of the component, truncated to the last 15 characters.
Computer Name	Up to 15	Name assigned to the machine hosting the process, truncated to the first 15 characters.
Type	1	A single character: I for Information, W for Warning, or E for Error.
Category	Up to 10	A component-specific text field. Components are not required to provide a value in this field.
Msg ID	Up to 8	A numeric message identifier that corresponds to a message string within a resource file associated with the calling component.
Message	Up to 1,024	The message text which may be either a string retrieved from a resource associated with the calling component or a text string passed directly from the calling component.

An entry may be followed by binary data in standard ASCII hexadecimal format. There is no limit to the length of the binary data.

Sample Entry

The following shows an example of the type of data contained in the event log.

Date	Time	Component	Process	Process Name
11-04-1997	14:01:05	NTMKDE	000000DD	W3DBSMGR.EXE

Computer Name	Type Category	Msg ID	Message
LABSERVER	I		MicroKernel is using default settings.

Error Code Clarification

In earlier Btrieve and SQL Interface releases, top-level components occasionally subsumed error codes from underlying components into an umbrella error code returned by the top level component. In some cases, this situation could make troubleshooting difficult because a single error code could have a wide range of possible root causes.

Starting with Pervasive.SQL 7, most top level components have been redesigned to pass through error codes from underlying components so that the actual source of the error is clearly identified to the calling application and/or in the log file.

In situations where an error code remains overloaded, specific information in the Pervasive.SQL event log should identify the root cause of the error.

Diagnosing Load Errors

Pervasive.SQL provides the following types of information you can use in diagnosing module load errors:

- Status codes. You can refer to the *Status Codes and Messages* manual for more information about the specific status code returned.

- Event log. You can look for the following information in the Pervasive.SQL Event Log to get additional information about a specific module load error:

Component Name

The logical or physical name of the module that received the load failure. Logical names used are:ServicesIfc—Abstract OS Services
BtrvIfc—Btrieve
MKDEIfc—MicroKernel
SSQLIfc—Scalable SQL

The physical name is logged if the calling module attempted to load a component using a binding rule. For example, a physical name is W3BIF102.

Type

The type of load error, as follows:

E (Error)—The module could not be found or an operating system-specific error occurred while loading the module.

W (Warning)—A symbol could not be found or was not exported by the module.

Message

The message depends on the type of module load error. If the module could not be found, the event log contains the binding rule that specified the downstream component. If an operating system-specific error occurred, the event log contains that operating system error. If a symbol could not be found or was not exported, the event log contains that symbol.

- On-screen errors. The event log is not functional until the Services DLL loads. Therefore, if a load error occurs while binding to the Services DLL, the event log does not log the error. Instead, Pervasive.SQL can display an on-screen error.

  You must enable the Services DLL to display on-screen module load errors by setting the PVSW_DISP_LOAD_ERRS environment variable. Its format is as follows:

```
PVSW_DISP_LOAD_ERRS=AIF
```

This environment variable should only be set to diagnose module load errors. In all other cases, it should not be set.

To diagnose an error, set this variable as specified and perform the operation. The load error is displayed in a message box on your system.

Configuring Components Using the Setup Utility

This chapter discusses the following topics:

- "Setup Utility Overview"
- "Special Notes on the Setup Utility"
- "Setup Utility Main Window"

Client-side components:

- "Scalable SQL Requester for Windows, Windows 95, and Windows NT Options"
- "DBNames Interface Options"
- "Btrieve Requester Options"
- "MicroKernel Router Options"
- "Communication Requester Settings"

Server and workstation components:

- "Scalable SQL Engine Options"
- "Scalable SQL Communications Manager Options"
- "MicroKernel Database Engine Options"
- "Btrieve Communications Manager Options"

Setup Utility Overview

The Setup utility manipulates settings for the Pervasive.SQL 7 workstation, client and server components.

Configuring these components is optional. If you do not configure them, each component loads with default configuration settings. The Setup utility works only with Pervasive.SQL 7 or later.

You can use the Setup utility for the following reasons:

- Your system or your Pervasive.SQL application requires you to adjust the settings. Refer to your application's documentation for recommended values. If you are running multiple applications concurrently, add the recommended values together. If you are running multiple applications sequentially, use the highest recommended value.

- You want to optimize the settings so that Pervasive.SQL provides the services you need without using more memory than necessary. (The stated memory requirements provide guidelines for optimizing your computer's resources.)

For your changes to take effect, you must shut down and then restart the Pervasive.SQL components. For more information, refer to *Getting Started* with Pervasive.SQL.

Special Notes on the Setup Utility

This section contains information to help you understand the Setup utility, which is described in the following sections:

- “Connecting to Different Environments”
- “Configuring Pervasive.SQL for Distributed Databases”
- “Performance Issues”
- “Archival Logging vs. Transaction Durability”
- “Interpreting Setting Parameters”

Connecting to Different Environments

The Setup utility can configure both local and remote environments depending on whether you have a server or workstation product. When you start the Setup utility, it is ready to configure a local environment.

To configure a remote environment, click the **Connect** button and enter the name of a server. To disconnect from a remote server, click the **Disconnect** button. When you are connected to a remote environment, you can view and change only server components. When using a Windows NT server as both a client and a server, the NT client can only be configured locally at the server.

Configuring Pervasive.SQL for Distributed Databases

You may configure Pervasive.SQL to run in a distributed environment such that the data dictionary resides on one server, and one or more data files reside on remote servers. To successfully operate in this environment, you must meet the following requirements:

- For full read and write access, your Scalable SQL or Btrieve servers must be version 4 or 7 or later, respectively. Otherwise, you can only perform read-only transactions (for example, if you are accessing data on a remote server running Scalable SQL 3.*x* or Btrieve 6.*x*).
- Your engine’s user name must be a valid user name across servers with the same password. On your primary server, the user must have administrative privileges.
- You must set the Accept Remote Requests setting of the Scalable SQL Communications Manager component to *On* .
- Scalable SQL must be loaded on the machine where the data dictionary files (DDF) reside and the machine where the data files are located. If only Btrieve is loaded on the machine where the data files reside, you will receive a Status Code 20, “The MicroKernel or Btrieve Requester is inactive.” To resolve this situation, load the Scalable SQL engine on the machine where the data files reside.

Performance Issues

This section contains notes on settings that can affect performance.

The configuration setting Cache Allocation has a default value of 1,024 KB. You may find that you get better performance with the Cache Allocation setting set to 2,048 KB or larger. If possible, set the cache size to the total size of all the Pervasive.SQL data files that will be accessed concurrently or the size of the largest data file that will be accessed sequentially.

Archival Logging vs. Transaction Durability

The Setup Utility allows you to specify two different logs that are written to the MicroKernel:

- Archival Logging
- Transaction Durability

The archival log facilitates the recovery of data files in case a system failure or crash occurs. (You can also use the roll forward feature in BUTIL or Btrieve Interactive Maintenance utilities to recover changes made to a data file between the time of the last backup and a system crash.)

A transaction durability log includes all transaction operations submitted to the MicroKernel from the time the application received the successful status from the END_TRANSACTION operation and any changes the engine wrote into the data file. See “Transaction Durability”

These two logs are never applied together. The archival log is used for restoring the current state from archives in case a hardware error or file corruption occurs. It can also be used for auditing purposes. The transaction log is used to provide better performance while ensuring that committed transactions do not get lost in case of a system crash.

For more information about Archival Logging or Transaction Durability, see Part I of the *Pervasive.SQL Programmer’s Guide* .

Interpreting Setting Parameters

Under each option, there are a number of specifications, which are displayed in the following table:

Range	The valid values the setting can take.
Default	The value that Pervasive.SQL assigns if you do not modify it.
Approximate memory required	If applicable, this value gives you an estimate of the memory price of using this option.
Applicable to clients	This field lists the Pervasive.SQL client requester versions that are appropriate to this option. If this field contains “Not Applicable,” this is a remote engine setting only and any Setup utility can be used to change the setting on a remote machine. If this field contains “Win32 only,” only Win32 Requesters are affected by this setting and you must use the Win32 Setup utility to change this option. If this field contains “Win16 only,” only Win16 Requesters are affected by this setting and you must use the Win16 Setup utility to change this option. If this field contains a list of clients, the setting applies to multiple requester versions and you must use the applicable Setup utility to change the option for each client you use. For example, use Win32 Configuration to change the setting for Win32 client requesters and the OS/2 Setup utility to change the setting for OS/2 client requesters.
Applicable to servers or workstations	This field lists the Pervasive.SQL server or workstation to which this setting applies, such as Windows NT or NetWare for the server, and Windows 9X/NT for the workstation.

The following table outlines the functionality of the three versions of the Setup utility:

Table 3-1
Pervasive.SQL Win16, Win32, and OS/2 Setup Utilities

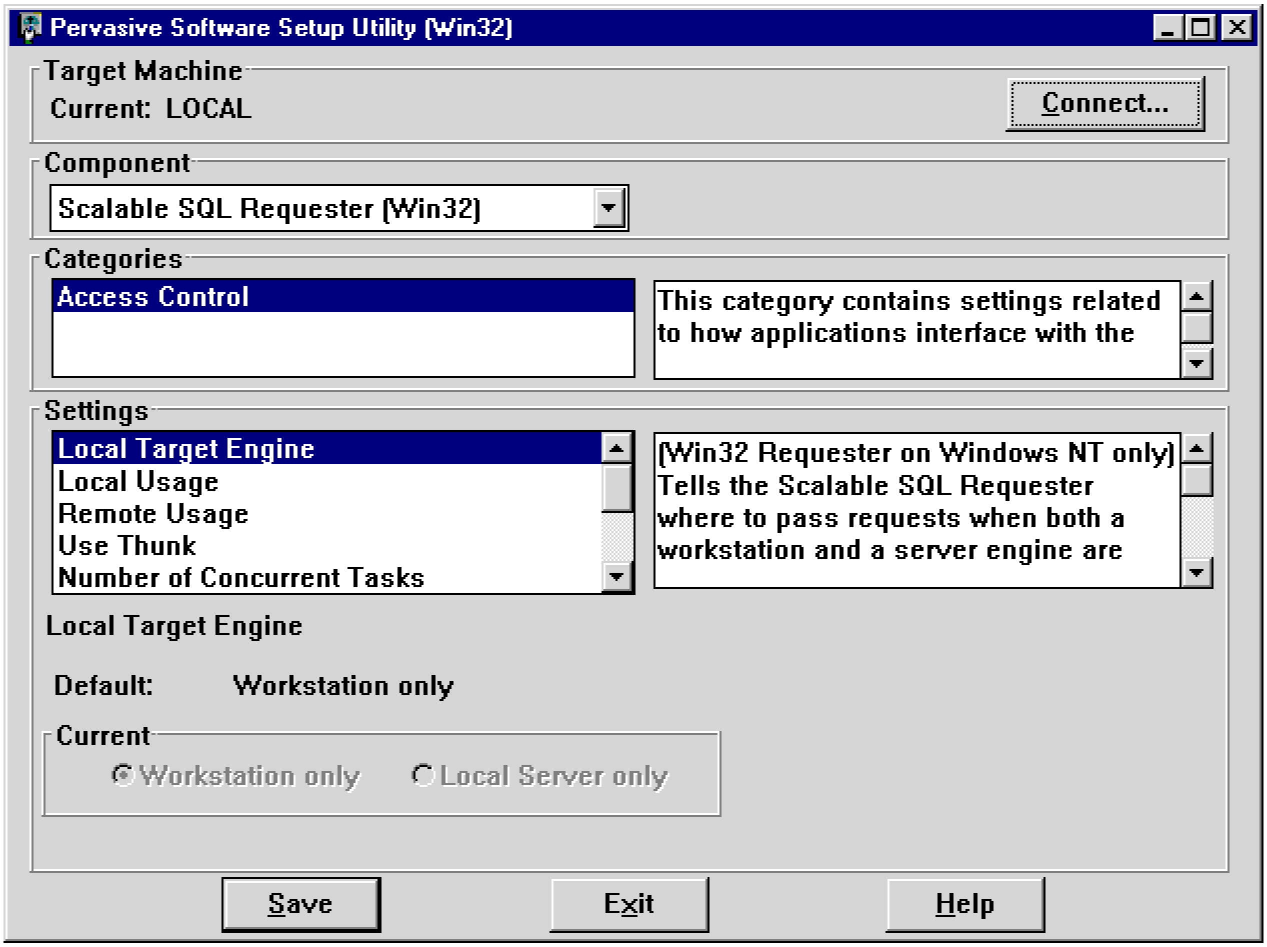
Setup utility	Components that it can configure	Where changed

Win32 Setup utility (Runs on Windows 9X/NT workstations or Windows NT servers.)	Local Win32 components	Windows Registry on local machine
NT Server components (local or remote machines)	Windows Registry	
NetWare Server components (remote machines only)	BTI.CFG on NetWare server	
Win16 Setup utility (Runs on Windows 3.x, Windows 9X/NT workstations and Windows NT servers.)	Local Win16 components NT Server components (remote machine only) NetWare Server components (remote machines only)	BTI.INI on local machine Windows Registry on Windows NT server BTI.CFG on NetWare server
OS/2 Setup utility (Runs on OS/2 client workstations)	Local OS/2 componentsNT Server components (remote machines only) NetWare Server components (remote machines only)	OS2.INI (binary) Windows Registry BTI.CFG

Setup Utility Main Window

Figure 3-1 shows the Setup utility's main window when it is connected to a remote server.

Figure 3-1
Setup Utility Main Window



Target Machine
Current: LOCAL

Component
Scalable SQL Requester (Win32)

Categories
Access Control

This category contains settings related to how applications interface with the

Settings
Local Target Engine
Local Usage
Remote Usage
Use Thunk
Number of Concurrent Tasks

[Win32 Requester on Windows NT only] Tells the Scalable SQL Requester where to pass requests when both a workstation and a server engine are

Local Target Engine

Default: Workstation only

Current
● Workstation only ○ Local Server only

Table 3-2 describes the elements in the Setup utility's main window.

Table 3-2
Elements in the Setup Utility Main Window

Configuration Element	Description
Target Machine	Displays LOCAL when configuring the local environment (a Windows 3.x, Windows 9X, or Windows NT client *or* a Windows 9X or Windows NT workstation) or a server name when configuring a remote environment.
Component	Lists the components available for configuration. **Note:** If a component is not loaded or not present, it does not appear in the list. Local components must be installed in a search path location; remote components must be loaded and running.

Categories

Lists the categories of configuration options for the current component.

Settings

Lists the configuration options you can change in the current category. Below the **Settings** list, the Setup utility displays the default and current settings for the highlighted option. The utility also displays the minimum and maximum values for the settings, where applicable.

Connect/Disconnect

Allows you to connect to or disconnect from a remote server.

Maintain Named Databases

Allows you to create bound databases. This button only appears when you select the Scalable SQL Engine component.

Scalable SQL Engine Options

This section describes the SQL engine configuration options in order by category:

- System Configuration
- Trace File Control

System Configuration

The System Configuration category includes six settings, which are described as follows:

Enable External Procedures

Range	Default	Memory Req	Clients	Engines
On or Off	Off	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting allows you to enable or disable Inscribe support. Inscribe lets you to create and run external procedures from a Pervasive.SQL database engine.

For the SQL server engine for the NetWare, if you select *Off* , the SQL engine does not make calls to Inscribe. If you select *On* , the SQL engine provides Inscribe support, allowing you to use external Inscribe procedures.

For the SQL engine for Windows NT server or Win32 workstation engines, if you select Off, the SQL engine does not load with Inscribe support. If you select On, the SQL engine loads with Inscribe support.

For more information about Inscribe, refer to the *Inscribe User's Guide* . For the Pervasive.SQL 7 Workstation, this manual is only available in a WinHelp file (INSCRIBE.HLP).

External Sort File Directory

Range	Default	Memory Req	Clients	Engines
Any valid directory path	NetWare: SYS:SYSTEM Windows NT: default Windows directory	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies where the SQL engine stores the temporary files it creates during certain processes. This directory must exist before the SQL engine accesses it and must be on the same server where the SQL engine is installed.

You can create a directory to hold the temporary files and then specify that directory as the external sort file directory. When specifying a path, use one of the following formats:

NetWare Format	Windows NT Format
vol :directory	*drive:directory*

The SQL engine assigns a coded name to each temporary file and deletes the temporary files upon completion of the processes that require those files. If you do not enter a path for this parameter, the SQL engine places the temporary files in the default directory of the server on which the SQL engine is installed.

Isolation Level

Range	Default	Memory Req	Clients	Engines
Cursor Stability or Exclusive	Cursor Stability	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting allows you to specify the engine's default isolation level (locking) for the SQL Interface. If you specify *Exclusive* , the default is exclusive, file-level locking; an exclusive lock is not released until the transaction is complete. Other users cannot access a file that another user has locked. If you specify *Cursor Stability* , the default is cursor stability locking, which locks either the row or page you are accessing. Other users cannot access a page or record that another user has locked.

You can use cursor stability locking only with 6.*x* or later data files.

Communications Buffer Size

Range	Default	Memory Req	Clients	Engines
4,096 through 32,767 bytes	16,384 bytes	Specified value *Maximum number of worker threads	N/A	Windows NT Server

This setting specifies the length in bytes of the longest block of data that can be transferred between a Windows NT application running on the server and the SQL engine for Windows NT. Each worker thread allocates a memory buffer large enough to accommodate this maximum length of data. Worker threads are the elements that actually perform file operations on behalf of the requesting client process.

If you set this value lower than the number of bytes your application requires, the SQL engine returns an error message at run time. However, setting a value higher than you need does not improve performance.

Worker Threads

Range	Default	Memory Req	Clients	Engines
2 through 64 threads	3 threads	N/A	N/A	Windows NT Server Windows 9X/NT workstations

This setting specifies the number of worker threads the SQL engine initially spawns to handle client requests. Worker threads are the elements that actually perform file operations on behalf of the requesting client process. (The SQL engine may dynamically spawn additional worker threads as needed.)

Check Table Definitions

Range	Default	Memory Req	Clients	Engines
On or Off	Off	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies whether the SQL engine checks the table definitions stored in the DDFs against the actual data file definitions. The SQL engine returns Status Code 353 when you attempt to query a table for which the data file definitions do not match the DDFs.

Trace File Control

The Trace File Control category includes five settings, which are described as follows:

Scalable SQL Logins/Logouts

Range	Default	Memory Req	Clients	Engines
On or Off	Off	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies whether to record database login and logout activity to a trace file. If you enable this option, you must specify a trace file using the Trace File option.

Failed Scalable SQL Logins

Range	Default	Memory Req	Clients	Engines
On or Off	Off	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies whether to record failed database login attempts to a trace file. If you enable this option, you must specify a trace file using the Trace File option.

All Other Scalable SQL Calls

Range	Default	Memory Req	Clients	Engines
On or Off	Off	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies whether to record all data definition and data manipulation activity to a trace file. Developers can use tracing to debug applications. If you enable this option, you must specify a trace file using the Trace File option.

Trace File

Range	Default	Memory Req	Clients	Engines
Any valid path	NetWare: directory from which the SQL engine is started + SSQL.LOG	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT

	Windows NT: directory in which the SQL engine is installed + SSQL.LOG			workstations

This setting specifies the file to which the SQL engine writes trace information. The path and file name must be valid.

> **Note:** Do not use the same trace file name for Scalable SQL and MicroKernel Database engines.

Maintain Named Databases

Range	Default	Memory Req	Clients	Engines
N/A	N/A	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

A named database has a logical name that allows users to identify it without knowing its actual location. When you name a database, you associate that name with a particular dictionary directory path and one or more data file paths. When you log in to the SQL engine using a database name, the SQL engine uses the name to find the database's dictionary and data files. A named database enables you to do the following:

- Define triggers
- Define primary and foreign keys
- Bind a database
- Suspend a database's integrity constraints

You use the **Maintain Named Database** feature to bind and name databases. Table 3-3 describes the elements in the Maintain Named Databases dialog.

Table 3-3
Elements in the Maintain Named Databases Dialog

Element	Description
Database Names	Lists the available named databases.
Dictionary Location	Displays the location of the DDF files for the selected database name.
Integrity Enforced	Displays whether Pervasive.SQL is enforcing integrity constraints (including security, referential integrity, and triggers) on the database.
Bound	Displays whether the database is bound. Binding a database ensures that the MicroKernel enforces the database's defined security, referential integrity (RI), and triggers, regardless of the method you use to access the data. For more information about bound databases, refer to the

	Pervasive.SQL Programmer's Guide.
Data File Locations	Displays the location of data files for the database.
Create	Allows you to create a new named database.
Modify	Allows you to change the properties of the current database.
Delete	Deletes the database name from the DBNAMES.CFG file. For bound databases, this operation also deletes the associated dictionary files. For unbound databases, this operation does not delete any files.
Bind	Makes the database a bound database. **Note:** To bind successfully, an existing database cannot reference dictionary or data files referenced by another named database. If your database does share dictionary or data files, you must ensure that all named databases sharing the dictionary files are unbound.
Unbind	Makes the database an unbound database and changes the status of each dictionary and data file to unbound. **Note:** Because the engine automatically stamps a file as bound if it has a trigger, a foreign key, or a primary key referenced by a foreign key, you may unbind the database, but have some files that remain bound. For more information about bound databases, refer to the *Pervasive.SQL* Programmer's Guide.
Check	Determines whether the entry for this database in the DBNAMES.CFG file accurately reflects the existing dictionary and data files.

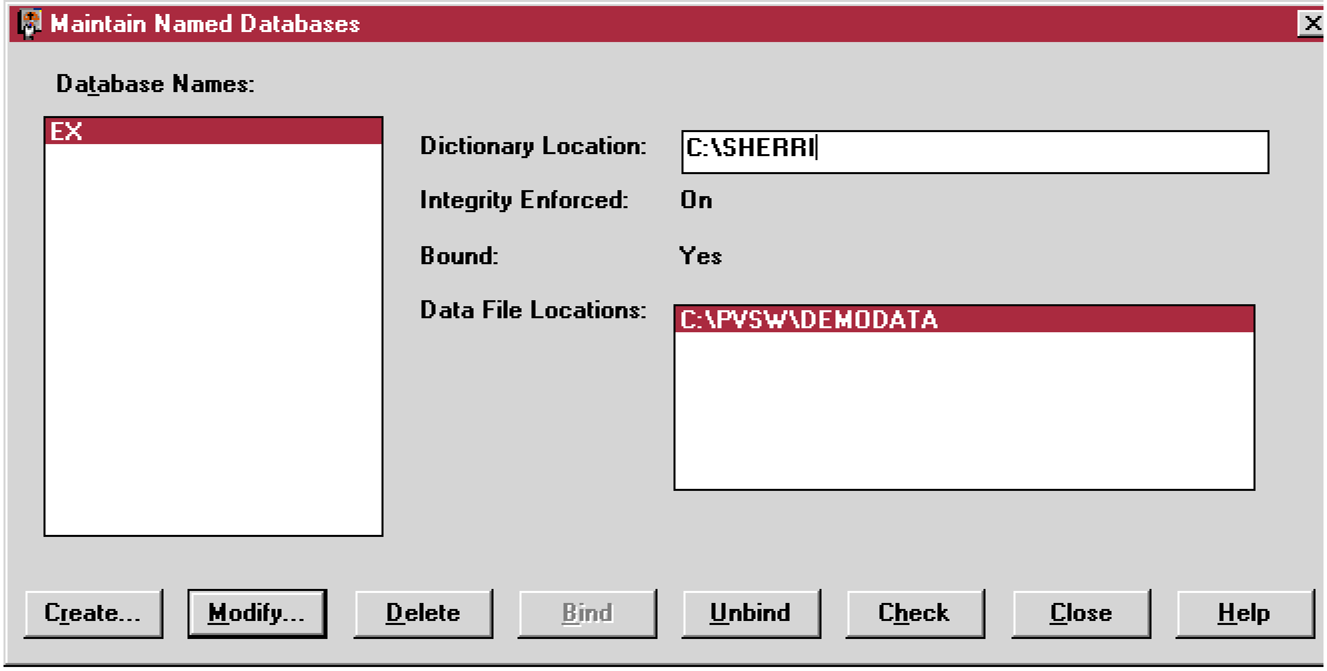
Creating New Bound Databases

To create a bound database:

1. From the Setup utility dialog, click **Maintain Named Databases** .

   The **Maintain Named Databases** dialog appears.

Figure 3-2
Maintain Named Databases Dialog



Maintain Named Databases

Database Names:

EX

Dictionary Location: C:\SHERRI

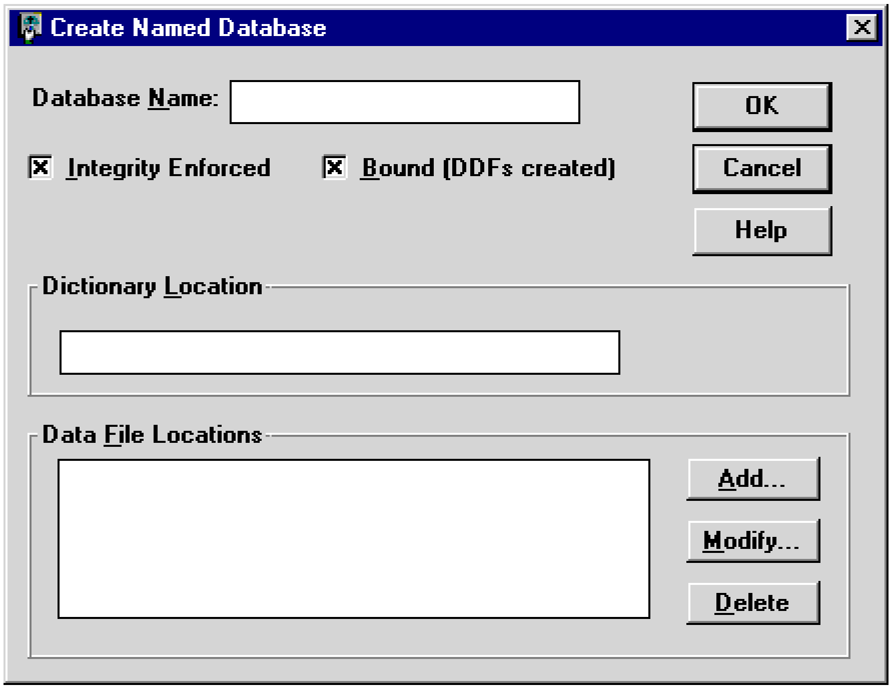
Integrity Enforced: On

Bound: Yes

Data File Locations: C:\PVSW\DEMODATA

2. Click **Create** . The **Create Database Name** dialog appears.

Figure 3-3
Create Named Database Dialog



Create Named Database

Database Name:

☒ Integrity Enforced ☒ Bound (DDFs created)

Dictionary Location

Data File Locations

3. In the **Database Name** field, enter the name of the database. Do not specify a database name that includes spaces. The SQL engine displays the name only up to the embedded space, so the name *My DB* is advertised as *My* .

4. By default, **Integrity Enforced** setting is enabled. This setting determines whether Pervasive.SQL enforces integrity constraints (including security, referential integrity, and triggers) on the database, regardless of whether the database is bound.

   In general, you should not disable integrity enforcement. However, you may want to suspend integrity constraints to facilitate bulk data loads.

5. By default, the **Bound** setting is enabled which determines that a database is bound. When you create a bound database, the SQL engine creates the appropriate dictionary files.

Note: You cannot create a bound database for which the dictionary files already exist. If you have existing dictionary files, first create an unbound database, then use the **Bind** button in the Maintain Named Databases dialog (see Figure 3-2).

6. Specify the dictionary location for the database in the **Dictionary Location** field. This location must be on the same server to which you are connected.

   For NetWare, enter a path in the form of vol:\path . For Windows NT, enter a path in the form drive:\path where drive should be a local drive letter.

7. Specify the location of the data file(s) in the **Data File Locations** field. Click **Add** to enter a location in the **Location** pop-up dialog.

   If you are specifying paths to data files on this server, specify paths the same as you did in the **Dictionary Location** field. If you are entering paths to data files on another server, specify the full name. For NetWare, specify the path in the form \\server\vol1\ path (UNC format) or server\vol1:\ path.. For Windows NT, specify the path in the form \\server\sharename\path .

   You can click **Delete** if you decide not include a specific data file location.

8. Click **OK** to create the named database and close the dialog, or click **Cancel** to close the dialog without saving changes.

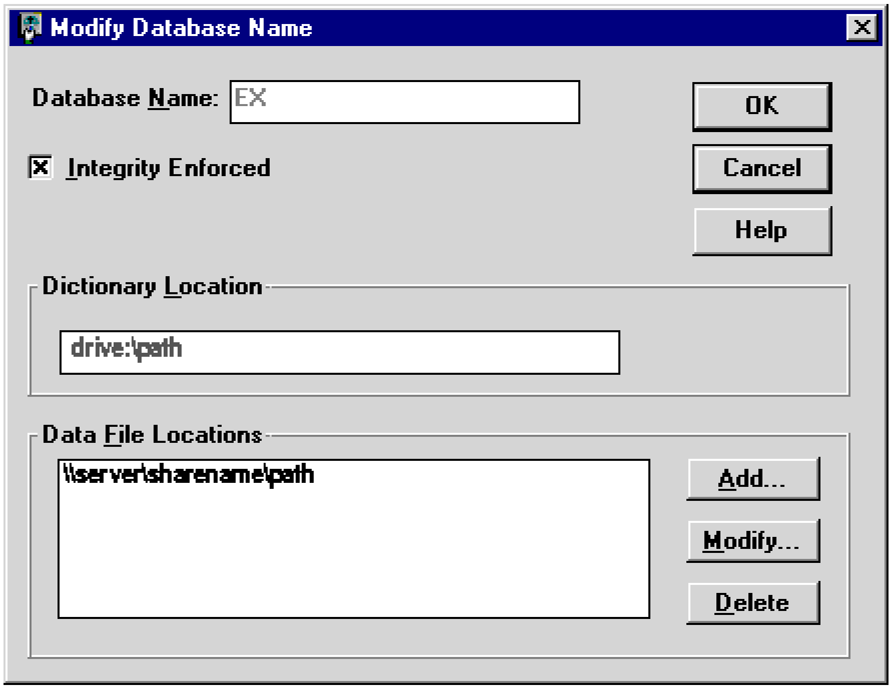
   The newly created named database and its data file locations are displayed in the **Maintain Named Databases** dialog. If you named a database but did not bind it, you can click the **Bind** button.

Modifying Named Databases

To modify a named database:

1. Either highlight the database name in the **Database Names** list and click **Modify** or double-click the database name. The following dialog appears.

Figure 3-4
Modify Database Name Dialog



Database Name: EX

☒ Integrity Enforced

Dictionary Location: drive:\path

Data File Locations: \\server\sharename\path

2. Add or modify data file locations.

> **Note:** You cannot modify the Database Name or Dictionary Location for bound databases or for databases that contain triggers or foreign keys.

3. Click **OK** to save your modifications. Otherwise, click **Cancel** to close this dialog without saving changes.

Deleting Named Databases

To delete a named database:

- Highlight the database name in the Database Names list and click **Delete** . The named database is removed from the list of database names.

Scalable SQL Communications Manager Options

This section describes the SQL Communications Manager configuration options for Pervasive.SQL server engines.

Server Communications Configuration

The Server Communications Configuration category includes seven settings, which are described as follows:

Number of Sessions

Range	Default	Memory Req	Clients	Engines
1 through 4,096 sessions	15 sessions	32 KB per session for Windows NT N/A for NetWare	N/A	WinNT and Netware only

This setting specifies the maximum number of network connections that can access the server engine at any given time. You cannot improve performance by specifying a value higher than you need. If you have multiple SQL applications running on one workstation, each application may generate one or more sessions to the SQL engine.

NetWare users only: The amount of memory this option uses impacts the memory required for the Receive Packet Size option. Those two options work together.

Communications sessions are related to SQL engine logins, but they are not the same. A SQL engine login always generates a new SQL engine session. Whether it generates a new communications session depends on the location of the database being logged into. If the same task logs into a database on the same server, the task has two different SQL engine sessions, but only one communications session. This is because the Requester reuses the network connection for the second login.

Communications Buffer Size

Range	Default	Memory Req	Clients	Servers
4,096 through 32,767 bytes (Windows NT) 512 through 65,116 bytes (NetWare)	16,384 bytes	*Communications Buffer Size* * (*Number of Communications Threads* + 1) + 385 bytes	N/A	WinNT and Netware servers

This setting specifies the size of the buffer (in bytes) that the SQL engine communications layer allocates for database requests from remote clients. This value should be at least as large as the largest data length parameter for your SQL Requesters.

> **Note:** *System Administrators* : Refer to the documentation for your Pervasive.SQL application to get an appropriate value for this option.
>
> *Application Developers* : If your application requires a communications buffer larger than the default, note the appropriate size in the documentation for your application.

Number of Communications Threads

Range	Default	Memory Req	Clients	Servers
2 through 64 threads (Windows NT) 1 through 200 threads (NetWare)	3 threads	*N/A*	N/A	Both

This setting specifies how many communications threads the SQL engine Communications Manager spawns to handle client requests. Communications threads are the server processes that actually perform file operations on behalf of the requesting client process. The number you should specify depends on the amount of communications activity. For light usage, the number can be smaller than the default. For heavy concurrent usage, use one thread per workstation, up to 64.

Accept Remote Requests

Range	Default	Memory Req	Clients	Servers
On or Off	On	*N/A*	N/A	Windows NT only

This setting is only used by the SQL engine for Windows NT; the SQL engine for NetWare ignores this option.

The setting specifies whether the Communications Manager accepts requests from remote servers and workstations. If you turn this option on, the Communications Manager advertises its presence on the network.

Supported Protocols

Range	Default	Memory Req	Clients	Servers
TCP/IP or SPXII	TCP/IP, if the protocol is available on both the client and the server. Both protocols are enabled by default, but TCP/IP is attempted first. If TCP/IP is not available, then SPXII is used.	*N/A*	N/A	Windows NT only

This setting specifies the protocols the Communications Manager uses. If you specify both protocols, the Communications Manager attempts to use TCP/IP first. If TCP/IP is not available, the Communications Manager uses SPXII.

Read Buffer Size

Range	Default	Memory Req	Clients	Servers
2 through 32,767 bytes	4,096 bytes	Amount of memory allocated for each currently active remote session	N/A	Windows NT only

This setting is only used by the SQL engine for Windows NT; the SQL engine for NetWare ignores this option.

The option specifies the size of the buffer (in bytes) that the SQL engine reads on packets from the operating system's communication layer. You set this value in bytes. Any value you enter is rounded up to the nearest multiple of the system page size (4 KB on Intel platforms) at the time that the engine allocates the buffer.

You should set this option equal to the Communications Buffer Size plus an allowance for system overhead (about 400 bytes). However, be aware that setting this option to a higher value than the default carries a memory penalty, because the system allocates a buffer of the specified size for each active remote client connection. For example, if you have 100 active remote clients and you have the buffer size set to 4 KB, the system allocates 400 KB of memory. If you have the buffer size set to 16 KB, the system allocates 1600 KB of memory.

Receive Packet Size

Range	Default	Memory Req	Clients	Servers
532 through 4,096 bytes	1,500 bytes	*(See the formula below.)*	N/A	NetWare only

Approximate Memory Required: The setting has a memory requirement that uses the following formula:
of receive packets * *Receive packet size* where *Number of receive packets* = whichever is greater of the following:
or (2 * *Number of sessions) or*
{*Number of sessions * (Communications buffer size / Receive packet size)* + 1}

This setting is only used by the SQL engine for NetWare; SQL engine for Windows NT ignores this option.

The option specifies the size of the individual network packets that this component receives. The default Receive Packet Size varies, depending on your network card and hardware capabilities. If you are using the Windows NT or Windows 95 SQL Requester on an Ethernet topology, use at least the default value for this option. If you are on a Token Ring topology, set this value to 4,096 bytes. Setting the value too low may result in workstation hangs or a Status Code 95, “The session is no longer valid.”

Scalable SQL Requester for Windows 3.x, Windows 9X, and Windows NT Options

This section describes configuration options for the Scalable SQL Requester for Windows, Windows 9X, and Windows NT platforms.

Access Control

The Access Control category includes 10 settings, described as follows:

Local Target Engine

Range	Default	Memory Req	Clients	Engines
Local Workstation only or Local Server Only	Server (default for server engine) Workstation only (default for workstation engine)	N/A	Win32 only	Windows NT Server Windows 9X/NT workstations

This option specifies which connection to use when more than one engine is available on the local machine. When an application attempts to access a local file, the SQL Interface checks the setting of the Target Engine option. If this option is set to Workstation only, the SQL Interface passes the request to the workstation, which processes the request and returns the appropriate information. If this option is set to Server, the SQL Interface passes the request to the local server engine. If the SQL Interface cannot find the target engine, you receive one of the following status codes:

- Status Code 2103 (if Remote Usage is enabled)
- Status Code 802 (if Remote Usage is disabled)

Local Usage

Range	Default	Memory Req	Clients	Engines
On or Off	On	Not applicable	Win16, Win32	Windows NT Server Windows 9X/NT workstations

This setting tells the Scalable SQL Requester whether the local workstation or the local Server Engine (on Windows NT) should be used to access a file. If you have the Pervasive.SQL server product, the local SQL engine runs on the client workstation of a user, not on a server.

Note: If you configure both Local Usage and Remote Usage settings to *On* , the client tries the remote engine first, then tries the local engine. It also tries Scalable SQL first, then tries Btrieve if all the requester settings are enabled.

Remote Usage

Range	Default	Memory Req	Clients	Engines
On or Off	On	N/A	Win16, Win32	Windows NT Server

				Windows 9X/NT workstations

This setting specifies whether the SQL Requester allows access to a server engine running on a remote server.

Use Thunk

Range	Default	Memory Req	Clients	Engines
Yes or NoOn or Off	No (for Server engine) On (for Workstation engine, 16 bit only)	N/A	Win16 only	Windows NT Server Windows 9X/NT workstations

This setting specifies whether the Win16 client requester uses thunking to access Win32 components. Thunking is the transition from executing 32-bit code to executing 16-bit code, or from executing 16-bit code to executing 32-bit code. This option can only be set using the Win16 Setup utility.

Number of Concurrent Tasks

Range	Default	Memory Req	Clients	Engines
5 through 2,000 tasks	10 tasks	Number of tasks * 20 bytes	Win16, Win32	Windows NT Server Windows 9X/NT workstations

This setting specifies how many tasks the SQL Interface can service at one time. (A single application can have multiple tasks.) This number is limited, in part, by the amount of memory available on the workstation.

The number of tasks you should specify depends on the number of SQL applications you anticipate using concurrently. Specifying a number of tasks that is much higher than you need wastes memory. Specifying a number that is lower than you need may keep some applications from accessing the SQL engine.

If you plan to use Microsoft Access to operate on SQL engine data, set this option to at least 15. Microsoft Access often generates multiple tasks.

Note: For Windows 3.*x* , this is the number of applications allowed to access Scalable SQL. For Windows NT and Windows 95, this is the number of threads allowed to access Scalable SQL from one application.

Number of Concurrent Sessions

Range	Default	Memory Req	Clients	Engines
3 through 2,000 sessions	10 sessions	(*Number of tasks * Number of sessions * bytes)*	*Win16, Win32*	*Windows NT Server* *Windows 9X/NT workstations*

This setting configures the maximum number of sessions the SQL engine allows per task. The number of sessions you can specify is limited, in part, by the amount of memory available on the workstation.

Communications sessions are related to SQL engine logins, but they are not the same. A SQL engine login always generates a new SQL engine session; whether it generates a new communications session depends on the location of the database being logged into. If the same task logs into a database on the same server, the task has two different SQL engine sessions, but only one communications session. This is because the Requester reuses the network connection for the second login.

If you plan to use Microsoft Access to operate on SQL data, set this option to at least 10. Microsoft Access often generates multiple tasks.

Note: For 16-bit applications, the total number of sessions (*number of tasks* * *number of sessions*) is limited internally to 2,000 sessions. Thunking doubles the amount of memory required, because the Win16 and Win32 DLLs must be loaded and their respective tables allocated.

Support Scalable SQL Callback Yield

Range	Default	Memory Req	Clients	Engines
On or Off	On	N/A	Win16 only	Windows NT Server Windows 9X/NT workstations

This setting applies to Win16 applications, even those running on a Win32 operating system.

This option specifies whether the SQL engine yields CPU support to other applications during a SQL Interface callback.

Support Enhanced XQLVersion

Range	Default	Memory Req	Clients	Engines
On or Off	On	N/A	Win16, Win32	Windows NT Server Windows 9X/NT workstations

This setting specifies whether to support the enhanced XQLVersion call added with the Scalable SQL 3.01.

Communications Buffer Size

Range	Default	Memory Req	Clients	Engines
4,096 through 32,767 bytes	32,767 bytes	N/A	Win16, Win32	Windows NT Server Windows 9X/NT workstations

This setting specifies the maximum data length transferable between the application and the SQL engine.

Local Convert/Mask

Range	Default	Memory Req	Clients	Engines
On or Off	On	N/A	W16 and W32	Windows NT Server Windows 9X/NT workstations

This setting specifies whether the APIs are processed by the SQL engine running on your server. If you are going to use any of the new SQL data types and you are configured to use a local Scalable SQL 3.01 engine, you must set this option to *On* . By default the SQL requesters for Windows, Windows NT, and Windows 95 handle certain calls such as XQLConvert and XQLMask locally. Setting this option to *Off* sends all API processing to the Scalable SQL 4.0 engine running on your server.

DBNames Interface Options

This section describes the Database Names Interface configuration options in order by category. These options apply to clients and are only applicable if you run the SQL engine in your environment.

Access Control

The Access Control category includes the following four settings:

Alternate Infobase

Range	Default	Memory Req	Clients	Engines
Any valid file path	N/A	N/A	Win32, Win16	Windows NT and NetWare Servers Windows 9X/NT workstations

Infobases aid network administrators by allowing workstations to use a centralized location for Database Names configuration information. Depending on your operating system, the Database Names Interface uses one of the following default infobases:

Windows NT and Windows 95	Registry on the machine that contains the entries used to configure the Database Names Interface
Windows 3.x	BTI.INI in the Windows directory

This setting specifies the path to an alternate infobase. The alternate infobase is a text file in the format of the BTI.INI and BTI.CFG files.

Transport

Range	Default	Memory Req	Clients	Engines
Named Pipes or Requester	Named Pipes	N/A	Win32, Win16	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies the transport mechanism to use to connect to the servers listed in the Scalable SQL Servers option. Use this option if you specified any server names for the Scalable SQL Servers option. If any Database Names are located on NetWare servers, specify Requester. Otherwise, specify Named Pipes.

Scalable SQL Servers

Range	Default	Memory Req	Clients	Engines
List of available SQL Interface servers	Not applicable	N/A	Win32, Win16	Windows NT and NetWare Servers

This setting lists the names of Windows NT and NetWare servers you want to query for Database Names. Use this option if your environment meets any of the following criteria:

- Your network has a Windows NT server that is advertising Database Names using a SAP Agent.

- Your network contains no NetWare servers providing Bindery Services.
- Your client machine uses the Microsoft client software.
- You are attempting to use a database name to access the SQL Interface via TCP/IP. (This applies to both Windows NT and NetWare servers.)

Named Pipes Read Timeout

Range	Default	Memory Req	Clients	Engines
0 - 65,535 milliseconds	500 milliseconds	N/A	Win32, Win16	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting is used only if the Transport option is set to Named Pipes. The option specifies the time in milliseconds that a Named Pipe read waits before timing out.

Btrieve Requester Options

This section describes the options for the Btrieve client requester.

Client Configuration

The Client Configuration category includes the following two settings:

Splash Screen

Range	Default	Memory Req	Clients	Engines
On or Off	On	N/A	Win32, Win16, OS2	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting controls whether or not the Btrieve Interface splash screen displays. The splash screen displays the first time a Btrieve Requester loads.

Note: The 32-bit Setup utility enables/disables the splash screen for 32-bit applications and 16-bit applications when thunking is set to *On* .
16-bit applications, if thunking is set to *Off* , use the 16-bit Setup utility to enable/disable the splash screen.

Check Parameters

Range	Default	Memory Req	Clients	Engines
On or Off	Off	N/A	Win16 Only	Windows NT and NetWare Servers

This setting controls whether the Win16 components verify their pointers. Use this option only during development.

MicroKernel Router Options

This section describes the MicroKernel Router configuration options in order by category.

Access Control

The Access Control category includes the following five settings:

Local

Range	Default	Memory Req	Clients	Engines
On or Off	On	N/A	Win32, Win16, OS2	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies whether to use the local engine to access data files.

Requester

Range	Default	Memory Req	Clients	Engines
On or Off	On	N/A	Win32, Win16, OS2	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies whether the MicroKernel Router allows access to a MicroKernel server engine running on a remote server.

Note: It is recommended that you keep the default Requester setting of *Yes* on Microsoft's File and Print Services for Netware (FPNW) servers running Pervasive.SQL. You may receive a Status Code 94, "The application encountered a permission error," if you change this setting to *No* when you are running the Btrieve Interface locally on the FPNW server and are using a local FPNW drive mapping or local FPNW UNC path.

Local Target Engine

Range	Default	Memory Req	Clients	Engines
Try Local Server, then Workstation, Local Server only, or Workstation only	Try Server, then Workstation for server engin Workstation only for workstation engine**1	N/A	Win32, OS2 only	Windows NT Server Windows 9X/NT workstations

#For Windows 9X platforms, this setting is disabled for the workstation engine and is set to Workstation only by default.

This setting specifies where the MicroKernel Router passes requests when both a workstation and a server engine are available.

> **Note:** Do not use the same trace file name for Scalable SQL and MicroKernel Database engines.

Use Thunk

Range	Default	Memory Req	Clients	Engines
Yes or No	Yes	N/A	Win16 only	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies whether the Win16 client requester uses thunking to access Win32 components. You must use the Win16 Setup utility to set this option.

Number of Load Retries

Range	Default	Memory Req	Clients	Engines
0 - 65,536 retries	5 retries	N/A	Win32, OS2 only	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies the number of times the MicroKernel Router attempts to connect to the target engine.

Communication Requester Settings

This section describes the settings for the Pervasive.SQL client-side Communications Requester.

Access Control

The Access Control category includes the following three settings:

Supported Protocols

Range	Default	Memory Req	Clients	Engines
TCP/IP or SPXII	Both	N/A	W32, W16, OS2	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies the protocols the Communications Requester uses. If you specify both protocols, the Communications Requester attempts to use TCP/IP first. If TCP/IP is not available, it uses SPXII.

TCP/IP Timeout

Range	Default	Memory Req	Clients	Engines
1 - 2147483687	15	N/A	W32, W16, OS2	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies the number of seconds the requester should wait for a TCP/IP connect request to succeed before timing out.

Runtime Server Support

Range	Default	Memory Req	Clients	Engines
Yes or No	Yes	N/A	W32, W16, OS2	Windows NT and NetWare Servers

This setting controls run-time server support. If enabled, the user name for the drive on which you are presently running will be used. Enter a user name and password, to be used for Runtime Server Support.

SUPERVISOR and ADMIN are not valid user names, even if supplied with the correct password. If the requester cannot find a login user name other than SUPERVISOR or ADMIN, there is no valid name to pass.

MicroKernel Database Engine Options

This section describes the MicroKernel configuration options in order by category:

- File Settings
- Memory Resources
- Client/System Transactions
- System Resources/Directories
- Trace Btrieve Operations
- NetWare Only Settings

File Settings

The File Settings category includes the following seven settings:

Open Files

Range	Default	Memory Req	Clients	Engines
1 through 64,000 files	50 files	1,024 bytes per file	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies the maximum number of unique files, including SQL Interface data dictionary files (.DDF), that can be open at one time on the server. This value determines the size of the internal tables used to track active files.

Handles

Range	Default	Memory Req	Clients	Engines
1 through *Limited by Memory*	200 handles	256 bytes per handle	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies the maximum number of logical file handles that the MicroKernel uses at one time. The number of handles is different from the number of Open Files. For example, if an application opens the same file twice, it has one file but two handles.

Index Balancing

Range	Default	Memory Req	Clients	Engines
On or Off	Off	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting controls whether the MicroKernel performs index balancing. Index balancing increases performance on read operations; however, when you enable this option, the MicroKernel requires extra time and may require more disk I/O during insert, update, and delete operations. For more information about index balancing, refer to Part III of the *Pervasive.SQL* Programmer's Guide.

Archival Logging of Selected Files

Range	Default	Memory Req	Clients	Engines
On or Off	Off	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting controls whether the MicroKernel performs archival logging, which can facilitate your file backup activities. If a system failure occurs, you can use the archival log files and the BUTIL -ROLLFWD or SQLUTIL -ROLLFWD command to recover changes made to a file between the time of the last backup and a system failure.

You must specify the files for which the MicroKernel is to perform archival logging by adding entries to an archival log configuration file you create on the volume that contains the files. For more information about archival logging, refer to the *Pervasive.SQL Programmer's Guide* .

Create File Version

Range	Default	Memory Req	Clients	Engines
7.x, 6.x, or 5.x	7.x	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

The 7.*x* MicroKernel can read files created in 5.*x* and 6.*x* versions of the MicroKernel. In addition, the 7.*x* version can write to files using the existing file format. In other words, it writes to 5.x files using the 5.x file format, writes to 6.x files using the 6.x file format, and writes to 7.x files using the 7.x file format.

This setting specifies the format in which all new files are created. Specify 5.*x* or v.6*x* only if you need backward compatibility with a previous version of the MicroKernel. Specifying 5.*x* or 6.*x* does not affect any existing 7.x files.

System Data

Range	Default	Memory Req	Clients	Engines
None, If needed, or Always	If needed	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

The MicroKernel uses system data to ensure transaction durability. The options are as follows:

- None. System data is not included on file creation.
- If needed. System data is added to the file on file creation if the file does not have a unique key.
- Always. System data is always added on file creation, regardless of whether the file has a unique key.

Even if a file has a unique key, you may want to include system data, because users can drop indexes. For more information about system data, refer to the *Pervasive.SQL Programmer's Guide* .

Maximum Databases

Range	Default	Memory Req	Clients	Engines
0 through the maximum limited by memory	10	800 bytes per database	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies the limit for the maximum number of databases this MicroKernel can open.

Memory Resources

The Memory Resources category includes the following five settings:

Cache Allocation

Range	Default	Memory Req	Clients	Engines
64 KB through Limited by Memory or 4,194,303 KB (in multiples of 16 KB)	1,024 KB	Size specified	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies the size of the cache (in kilobytes) that the MicroKernel allocates; the MicroKernel uses this cache for all data file accesses. The MicroKernel uses options that are multiples of 16 KB. If you specify a number that is not a multiple of 16 KB, the MicroKernel rounds that number down to the nearest multiple of 16 KB. This number is multiplied by the number of I/O threads.

To achieve best performance, allocate a cache size no larger than the sum of the sizes of the files you are using. However, be careful not to take all available cache, especially when the server is running other applications. You cannot improve performance—and may waste memory—by specifying a value higher than you need.

Communication Buffer Size

Range	Default	Memory Req	Clients	Engines
1 - 64 KB	16 KB	*Number of workers threads* * (*Comm buffer size* + 400)	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

To calculate the required memory for this option, you need to know the settings of your Worker Threads and Communication Buffer Size options.

This setting specifies the length in bytes of the longest block of data that can be transferred between an application running on the server and the MicroKernel server engine. Each worker thread allocates a memory buffer large enough to accommodate this maximum length of data. (A message is a unit of related data that the MicroKernel or the application passes over the network.)

Note: *System Administrators* : Refer to the documentation for your Pervasive.SQL application to get an appropriate value for this option. If you use multiple applications, use the largest value.
Developers : If your application requires a communications buffer larger than the default, note the appropriate size in the documentation for your application.

Setting a value higher than you need does not improve performance.

Largest Compressed Record Size

Range	Default	Memory Req	Clients	Engines
0 - 64,000 KB	5 KB	2,048 bytes * *Largest compressed record size*	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies the size of a compression buffer that the MicroKernel uses when you access records in a file created with the Data Compression file attribute enabled. The MicroKernel allocates a compression buffer with a size of 2,048 bytes multiplied by the value you specify for this option. (You can determine this specified value by either the record size or the page size.)

Use the following guidelines when specifying the value for this option:

- If you use compressed files, determine the size (in bytes) of the largest record in any of your compressed files. Round any uneven values up to the next whole kilobyte.

  For example, if the largest record you will access is 1,800 bytes, specify a value of 2 for this option. The MicroKernel allocates 4,096 bytes (that is, 2,048 * 2) of memory for its compression buffer.

- If you do not use compressed files but you have at least one 7.0 file with a record length greater than 4,076 bytes, set this value to at least 5.

- If you do not use compressed files and you have no files with a record length greater than 4,076 bytes, set this value to 0. You cannot improve performance—and may waste memory—by specifying a value higher than you need.

Extended Operation Buffer Size

Range	Default	Memory Req	Clients	Engines
0 - 64,000 KB	16 KB	Size specified	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies the size (in kilobytes) of the buffer required to handle extended (multiple record) operations.

Sort Buffer Size

Range	Default	Memory Req	Clients	Engines
0 KB through memory limitation	0 KB	Size specified	N/A	Windows NT and NetWare Servers

of your system or 4,194,303 KB				Windows 9X/NT workstations

This setting specifies the maximum amount of memory (in kilobytes) that the MicroKernel dynamically allocates and deallocates for sorting purposes during run-time creation of indexes. If the memory required for sorting exceeds the size specified or is greater than 60 percent of the available process memory, the MicroKernel creates a temporary file. The amount of available memory for a process is a dynamic value and varies according to system configuration and load. If you specify 0, the MicroKernel allocates as much memory as needed, up to 60 percent of the available memory.

Client/System Transactions

Transaction Durability

Range	Default	Memory Req	Clients	Engines
On or Off	On	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting controls whether the MicroKernel performs transaction durability by logging all transactional operations to a single transaction log. A transactional operation is a set of operations that occur after a Begin Transaction (19) and before either an End Transaction (20) or Abort Transaction (21). Transaction durability only applies to transactional operations.

Transaction durability is the assurance that the MicroKernel finishes writing to the log before returning a successful status code. This option also guarantees transaction atomicity, which ensures that if a given statement does not execute to completion, then the statement does not leave partial or ambiguous effects in the database.

Even when you turn Transaction Durability on, some files may not be transaction durable. A file must contain at least one unique key. For files that do not contain a unique key, you can use a system-defined log key. For more information about transaction durability and system data, refer to the *Pervasive.SQL Programmer's Guide* .

Operation Bundle Limit

Range	Default	Memory Req	Clients	Engines
1 - 65,535 operations	1,000 operations	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This option specifies the maximum number of operations (performed on any one file) required to trigger a system transaction. The MicroKernel initiates a system transaction when it reaches the bundle limit or the Initiation Time Limit, whichever comes first, or when it needs to reuse cache.

The MicroKernel Database Engine treats each user transaction (starting with Begin Transaction until End Transaction or Abort Transaction) as one operation.   For example, if there are 100 Btrieve operations between the Begin Transaction and the End Transaction operation, then all the 102 Btrieve operations together will be treated as one single operation.

Initiation Time Limit

Range	Default	Memory Req	Clients	Engines
1 - 65,535 milliseconds	1,000 milliseconds	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies the time limit (in milliseconds) that triggers a system transaction. The MicroKernel initiates a system transaction when it reaches the Operation Bundle Limit or the time limit, whichever comes first, or when it needs to reuse cache.

Log Buffer Size

Range	Default	Memory Req	Clients	Engines
64 KB through the upper limit of your system memory	64 KB	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies the size (in kilobytes) of both the transaction log buffer and the archival log buffer that the MicroKernel uses. You can enhance performance by increasing the log buffer size, because the MicroKernel writes the log information to disk less frequently.

Note: If you set the Log Buffer Size to a value greater than that of your Transaction Log Size, then the MicroKernel automatically increments the Transaction Log Size to the value you specified for the Log Buffer Size.

Transaction Log Size

Range	Default	Memory Req	Clients	Engines
64 KB through smallest of the following: –Available disk space –Operating system file limit –4,096 MB	512 KB	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies the maximum size of a transaction log segment. When the log file reaches its size limit, the MicroKernel closes the old log segment file and starts a new one. You might want to limit the size of your transaction log segments, as this reduces the disk space that the MicroKernel uses temporarily. However, limiting the size of the transaction log segments does entail more processing by the MicroKernel, because it has to close and create log segments more frequently. This can decrease performance.

> **Note:** If you set the value for this option less than the value you specified for the Log Buffer Size, the MicroKernel Database Engine automatically adjusts the Transaction Log Size by setting it to the value of the Log Buffer Size option.

System Resources/Directories

The System Resources/Directories category includes the following 11 settings:

Active Clients

Range	Default	Memory Req	Clients	Engines
1 - 65,535 clients	30 clients	250 bytes per client	N/A	Windows NT and NetWare Servers

This setting specifies the maximum number of clients that can access the MicroKernel at one time.

> **Note:** You cannot improve performance by specifying a value higher than you need. However, if you receive Status Code 162, “Client table is full,” increase the value for this option.

I/O Threads

Range	Default	Memory Req	Clients	Engines
1 - 128 threads	4 threads	8 KB per thread	N/A	Windows NT and NetWare Servers

This setting specifies how many background I/O threads the MicroKernel spawns. These threads perform disk I/O on a file and manage the MicroKernel’s cache. When the MicroKernel updates or writes to data files, it assigns each file to a particular I/O thread sequentially. When it reaches the last thread, the MicroKernel starts over until all data files have been assigned to a background thread. Because the MicroKernel does not spawn additional I/O threads as needed, specify the maximum number of I/O threads you anticipate needing.

For best performance, set this value to the number of *Open Files* or *8* , whichever is less. Specifying a value higher than 8 may degrade performance. Do not set this value higher than the number of Open Files.

> **Note:** *Application Developers:* There is no accurate way to calculate the appropriate number of I/O threads as this setting depends on the machine’s characteristics, OS configuration, and the MicroKernel Database Engine’s planned work load.

Wait Lock Timeout

Range	Default	Memory Req	Clients	Engines
0 - 4,294,967 seconds	30 seconds	N/A	N/A	Windows NT and NetWare Servers
(49.7 days)				Windows 9X/NT

workstations

This setting specifies the wait lock timeout for the MicroKernel. When you fetch records with a wait lock, the MicroKernel does not return control until it has obtained the lock on every record you requested. If another application has locked one of the records you requested, the MicroKernel waits until that application releases the record before proceeding with the lock request. If the wait lock timeout has been reached and the MicroKernel could not lock the record, the MicroKernel returns control to its caller with the appropriate status code.

The purpose of this option is to significantly reduce network traffic, therefore improving network performance in case of a conflict caused by locking. With one exception (as stated in the following note), this configuration option does not have any effect on your application if there is a requester (such as W3BIFxyy.DLL) between your application and the MicroKernel. In this case, even if the wait lock timeout is reached, the requester retries the operation (except for Win16 applications) without notifying your application. The control is returned to your code only if the lock has been granted or a deadlock has been detected.

Note: If you have Win16 applications working with Pervasive.SQL, you may want to set this option to a value lower than the default (such as 1 second). For more information on how the MicroKernel handles wait locks with Win16 applications running on Windows 3.*x* , Windows 95, or Windows NT, refer to the *Pervasive.SQL Programmer's Guide* .

System Cache

Range	Default	Memory Req	Clients	Engines
On or Off	On	N/A	N/A	Windows NT Server Windows 9X/NT workstations

This setting is only used by the Windows NT MicroKernel; the NetWare MicroKernel ignores it.

The option specifies whether the MicroKernel should use the Windows NT system cache in addition to the MicroKernel's own Cache Allocation. In most cases, performance is enhanced by turning on the System Cache. However, if your computer's total memory is relatively small, you can turn off this option. You can use the Paging File and Process objects in the Windows NT Performance Monitor utility to determine whether the Windows NT system cache is being used effectively. For the NTDBSMGR instance, monitor the % Usage and % Usage Peak in the Page File object and the Page Faults/Second and Page File Bytes in the Process object.

Note: *Application Developers:* If you are experiencing poor performance and the available bytes are low and stay low (especially during activity with the MicroKernel), turn off the System Cache. However, if you turn this setting off, your application must meet the following requirements:

– File access must begin with multiples of the volume's sector size;

– File access must be for the number of bytes that are multiples of the volume's sector size. For example, if the sector size is 512 bytes, an application can request reads and writes of 512, 1024, or 2048 bytes, but not 335, 981, or 7171 bytes; and

– Buffer addresses for read and write operations must be aligned on addresses in memory that are multiples of the volume’s sector size.

Allocate Resources At Startup

Range	Default	Memory Req	Clients	Engines
On or Off	Off	N/A	N/A	Windows NT Server Windows 9X/NT workstations

This setting instructs the MicroKernel to allocate resources, including threads and memory buffers, when the MicroKernel is started. If you turn this option off, the MicroKernel does not allocate any resources until the first operation request. Pervasive.SQL applications automatically allocate resources as needed. Therefore, in most cases you do not need to do so explicitly.

Back To Minimal State If Inactive

Range	Default	Memory Req	Clients	Engines
On or Off	On	N/A	N/A	Windows NT Server Windows 9X/NT workstations

This setting causes the MicroKernel to free considerable memory and thread resources to the system and return to a minimal state when there are no active clients. (This is the initial state in which the MicroKernel begins.) The MicroKernel reallocates resources when another client becomes active.

Minimal State Delay

Range	Default	Memory Req	Clients	Engines
0 - 4,294,967 seconds (49.7 days)	30 seconds	N/A	N/A	Windows NT Server Windows 9X/NT workstations

This setting specifies a time interval for the MicroKernel to wait before returning to a minimal state. (This is the initial state in which the MicroKernel begins.) By returning to a minimal state, the MicroKernel frees considerable memory and thread resources to the system. In some cases, you may not want the MicroKernel to return to a minimal state. For example, you may be running a batch file that uses the MicroKernel repeatedly. The MicroKernel reallocates resources when another client becomes active.

Note: This setting is ignored if the Accept Remote Requests option is set to *Off*

.

Worker Threads

Range	Default	Memory Req	Clients	Engines
1 - 128 threads	1 thread	*Worker Threads* * (*Maximum record size* + 400)	N/A	Windows NT Server Windows 9X/NT workstations

This setting specifies how many worker threads the MicroKernel initially spawns to handle client requests. Worker threads are the elements that actually perform file operations on behalf of the requesting client process. The MicroKernel may dynamically spawn additional worker threads as needed to handle operation requests when all other workers are busy.

Note: These worker threads serve local clients only. A separate pool of threads serves remote clients.

Windows NT developers: There are two Worker Threads parameters: one under the System Resources/Directories category of the MicroKernel Database Engine component and one under the Server Communication Configuration category of the Btrieve Communications Manager component.

Worker threads for the **System Resources** category applies to local clients only (for example, applications running on Windows NT servers where the MicroKernel is running). This setting determines the number of worker threads the MicroKernel Database Engine should initially spawn during startup to handle local requests. The MicroKernel dynamically spawns additional worker threads when necessary to perform file operations for the local client(s).

The default setting is 1. If local applications require this setting to be higher, the MicroKernel can initially be configured to allocate the proper number of threads at startup. This may improve performance because the MicroKernel does not need to take the time to spawn additional threads dynamically after the applications have been started.

You use the NT Performance Monitor to verify the maximum number of threads the MicroKernel needs when processing local clients' requests. From the Performance Monitor, choose **Thread** and then look for the number of NTMKDE threads displayed in the Instance field.

Note: The MicroKernel by default starts with six total threads (five MicroKernel threads plus one worker thread). If the total number of threads is more than six (after running the local applications), the MicroKernel used more than one worker thread. To calculate the maximum number of worker threads needed for the local clients, subtract five from the total number of NT MicroKernel Database Engine threads. You can use the Pervasive.SQL Setup utility to modify the System Resources worker threads.

The Number of Communications Threads setting of the Btrieve Communications Manager component has the same purpose as the System Resources worker threads. These threads are used to handle file operation for remote clients (workstation applications). However, Communications Server threads do not get spawned dynamically.

Although it may be optimal to have one thread for each remote session, this is not feasible in most multi-user environments. According to Microsoft guidelines on the architecture used in the Btrieve Communications Manager, starting more than 2-3 worker threads per processor will not provide significant gain in performance and eventually could make the overall performance worse.

Home Directory

Range	Default	Memory Req	Clients	Engines
Any valid fully-qualified path	Directory from which the engine was loaded	N/A	N/A	Windows NT Server Windows 9X/NT workstations

This setting specifies the location the MicroKernel uses to store some system files and uses as a default location for other items, such as temporary work files. The path must include a drive specification or a UNC path.

Note: You can use the BTRINTF environment variable to temporarily override the home directory specification, as follows:

SET BTRINTF=/H: drive:\ path
drive is a drive letter and path is the path for the home directory.

Transaction Log Directory

Range	Default	Memory Req	Clients	Engines
Any valid fully-qualified path	Directory from which the engine was loaded	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies the location the MicroKernel uses to store the transaction log. The path must include a drive or volume specification or UNC path. The directory must exist.

Working Directory

Range	Default	Memory Req	Clients	Engines
Any valid fully-qualified path	None	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies the location of the MicroKernel working directory, which is used to store temporary files in operations such as building large repeating-duplicatable indexes. If disk space is limited on certain volumes, you can use this option to specify a working directory on a volume with adequate space. To specify a working directory, enter the path in the Current text box. The path must include a drive or volume specification or a UNC path. If you do not specify a working directory, the default is the location of the data file.

Trace Btrieve Operations

The Trace Btrieve Operations category includes the following eight settings:

Trace Operations

Range	Default	Memory Req	Clients	Engines
On or Off	Off	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting enables or disables the trace feature, which allows you to trace each Btrieve API call and save the results to a file. Developers can use tracing to debug applications. The MicroKernel writes to the trace file using forced write mode, which ensures that data gets written to the file even if the MicroKernel unloads abnormally. The MicroKernel's performance can be severely impacted, depending on the frequency of incoming requests. If you enable this option, you must specify a Trace File.

Trace File

Range	Default	Memory Req	Clients	Engines
Any valid fully-qualified file name	NT: the directory from which the engine was loaded NetWare: sys:\system\mkde.tra	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies the trace file to which the MicroKernel writes trace information. The file name must include a drive or volume specification and path or use a UNC path.

Note: Do not use the same trace file name for Scalable SQL and MicroKernel Database engines.

Select Operations

Range	Default	Memory Req	Clients	Engines
Any valid Btrieve operation code	All	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

The Available Operations scrollable list displays the available Btrieve Interface operation codes that you can trace. To choose the code(s), select it from the list and then click the **Add** button. To add all of the Btrieve Interface operation codes, click the **Add All** button. The added operation code appears in the Traced Operations scrollable list.

You can remove a Btrieve Interface operation code or codes from the Traced Operations list by selecting it and clicking the **Del** button. To remove all the Btrieve Interface operation codes, click the **Del All** button. The removed operation code appears in the Available Operations scrollable list.

Number of Bytes from Data Buffer

Range	Default	Memory Req	Clients	Engines
0 - 65,535 bytes	32 bytes	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies the size of the data buffer that the MicroKernel writes to the trace file when you enable the tracing feature to write to a file. The size you specify depends on the nature of your tracing needs (whether you need to see the entire data buffer contents or just enough of the buffer contents to identify a record).

Number of Bytes from Key Buffer

Range	Default	Memory Req	Clients	Engines
0 - 255 bytes	32 bytes	N/A	N/A	Windows NT and NetWare Servers Windows 9X/NT workstations

This setting specifies the size of the key buffer that the MicroKernel writes to the trace file when you enable the tracing feature to write to a file. The size you specify depends on the nature of your tracing needs (whether you need to see the entire key buffer contents or just enough of the buffer contents to identify a key).

NetWare Only Settings

This category contains settings available only on NetWare. These include BROUTER and Runtime Server Support settings.

Load BROUTER

Range	Default	Memory Req	Clients	Servers
On or Off	Off	N/A	N/A	NetWare only

This setting controls whether the Message router (BROUTER.NLM) is loaded during the execution of the BSTART command. The Message Router allows other applications running as NLMs on the server (such as the SQL Interface) to communicate with remote servers on which the MicroKernel is loaded. To access data on a remote server, set this option to *On* .

BROUTER Communications Buffer Size

Range	Default	Memory Req	Clients	Servers
1 - 64 KB	16 KB	(*BufferSize* + 355 bytes) * 4	N/A	NetWare only

This setting specifies the maximum length of the user data that any local server MicroKernel application can access at a remote server via BROUTER. Specify the length of the user data in bytes. Specifying a value higher than you need does not improve performance and may waste memory.

Runtime Server Support

Range	Default	Memory Req	Clients	Servers

Complete, Disabled, or Pre-Authorized	Complete	N/A	N/A	NetWare only

This setting specifies the level of Runtime Server Support provided by the MicroKernel.

When you specify **Complete** , a user is required to provide a valid user name; you can also specify a password, although not required.

If you specify **Pre-Authorized** , you are required to enter a valid user name and password. When you specify **Disabled** , you must have a connection to the NetWare server to access any Btrieve files. If you are not connected to the server, you will receive a Status Code 99, “The Btrieve Requester is unable to access the NetWare Runtime server.”

SUPERVISOR and ADMIN are not valid user names, even if supplied with the correct password. If the Requester cannot find a login user name other than SUPERVISOR or ADMIN, there is no valid name to pass. For more information about Runtime Server Support, see your Novell documentation.

Btrieve Communications Manager Options

This section describes the Btrieve Communications Manager configuration options for Pervasive.SQL servers.

Server Communication Configuration

The Server Communication Configuration category includes the following eight settings:

Number of Sessions

Range	Default	Memory Req	Clients	Servers
Windows NT: 0 through upper limit of your system memory NetWare: 1 - 4,906	15 sessions	Windows NT: 32 KB per session	N/A	Both

This setting specifies the maximum number of network connections that can access the server at any given time. You cannot improve performance by specifying a value higher than you need. The amount of memory this option uses impacts the memory required for the Receive Packet Size option; the two of these options work together.

If you have multiple applications running on one client, each application may generate one or more sessions to the MicroKernel. On NetWare, you can have the same number of sessions for each protocol. For example, if you set this to 10, you can have 10 SPX and 10 TCP/IP sessions, making a total of 20 sessions.

Communications Buffer Size

Range	Default	Memory Req	Clients	Servers
Windows NT: 512 - 65,116 bytes NetWare: 1 - 63 KB	Windows NT: 16,384 bytes NetWare: 16 KB	*Comm buffer size* * (*Number of Communications Threads* + 1)	N/A	Both

This setting specifies the size of the buffer (in bytes) that the Btrieve communications layer allocates for database requests from remote clients. This value should be at least as large as the largest data length parameter for your Btrieve Requester.

Number of Communications Threads

Range	Default	Memory Req	Clients	Servers
Windows NT: 1 - 128 threads NetWare: 1 - 200	3 threads	8 KB per worker thread	N/A	Both

This setting specifies how many communications threads the Btrieve Communications Manager dynamically spawns to handle client requests. Communications threads are the server processes that actually perform file operations on behalf of the requesting client process. On NetWare, each supported protocol spawns the number of communications threads.

Accept Remote Requests

Range	Default	Memory Req	Clients	Servers
On or Off	On	N/A	N/A	Windows NT

This setting specifies whether the Communications Manager accepts requests from remote servers and client workstations. If you turn this option to *On* , the Communications Manager advertises its presence on the network.

Supported Protocols

Range	Default	Memory Req	Clients	Servers
TCP/IP and SPXII	Both	N/A	N/A	Both

This setting specifies the protocols the Communications Manager uses. If you specify both protocols, the Communications Manager attempts to use TCP/IP first. If TCP/IP is not available, the Communications Manager uses SPXII.

Read Buffer Size

Range	Default	Memory Req	Clients	Servers
4 - 64 KB	4 KB	Amount of memory allocated for each currently active remote session	N/A	Windows NT

This setting specifies the size of the buffer (in kilobytes) that the MicroKernel reads on packets from the operating system's communication layer. You set this value in kilobytes. Any value you enter is rounded up to the nearest multiple of the system page size (4 KB on Intel platforms) at the time that the engine allocates the buffer.

You should set this option equal to the Communications Buffer Size plus an allowance for system overhead (about 400 bytes). However, be aware that setting this option to a higher value than the default carries a memory penalty, because the system allocates a buffer of the specified size for each active remote client connection. For example, if you have 100 active remote clients and you have the buffer size set to 4 KB, the system allocates 400 KB of memory. If you have the buffer size set to 16 KB, the system allocates 1600 KB of memory.

NetWare Settings Only

Receive Packet Size

Range	Default	Memory Req	Clients	Servers
532 - 4,096 bytes	1,500 bytes	(See the following formula.)	N/A	NetWare

Approximate Memory Required:

As indicated in the Memory Required field, this setting has the following formula:

Number of receive packets * *Receive packet size*
where *Number of receive packets* = (*Communications Buffer Size* / *Receive packets*) + 1, or 45, whichever is greater

This setting only applies to the SPX protocol and specifies the size of the individual network packets this component receives. For the approximate memory required, the number of receive packets can grow dynamically during execution, but starts with the number indicated.

The default Receive Packet Size varies depending on your network card and hardware capabilities. If you are using the Win32 client Requester on an Ethernet topology, use the default value for this option. If you are on a Token Ring topology, set this value to 4,096 bytes. Setting the value too low may result in workstation hangs or a Status Code 95, “The session is no longer valid.”

Use SAP

Range	Default	Memory Req	Clients	Servers
Auto Detect, Yes, No	Auto Detect	N/A	N/A	NetWare

This setting specifies whether the Btrieve Communications Manager should use the Service Advertising Protocol (SAP). This setting applies to SPX communications only.

Creating and Maintaining DDFs with DDF Ease

This chapter provides detailed information about creating and maintaining DDFs for existing Btrieve data files. This chapter includes the following sections:

DDF Ease Overview

This section introduces DDF Ease and provides conceptual information about data dictionary files.

What are DDFs

Data Dictionary Files (DDFs) describe data in your Btrieve database in terms of tables, columns, and indexes (in Btrieve terminology, this is Files, Fields, and Indexes).

A Btrieve database does not explicitly contain information that describes the format and meaning of the data in the database (this information is defined within a Btrieve application). DDFs provide a way of defining the fields in the database so that ODBC, Scalable SQL, and a variety of other commercial tools and applications can access the data in your Btrieve database. The format used to describe the structure and meaning of the data changed from Scalable SQL 3.01 to Scalable SQL 4.

What is DDF Ease

DDF Ease is a Win32 application that allows Pervasive database developers to create and maintain Data Dictionary Files (DDFs) and database files. With DDF Ease, you can add relational capabilities to an existing Btrieve navigational database, create new databases, design new tables, check your database for inconsistencies with table definitions, and convert your dictionary from Scalable SQL 3.x to Scalable SQL 4.x dictionary format. DDF Ease performs Btrieve calls to obtain table statistics and other information. This means that you should have at least read permission on the server.

DDF Ease creates and maintains DDFs that are standard to Scalable SQL and ODBC. These DDFs are FILE.DDF, FIELD.DDF, and INDEX.DDF.

Because DDF Ease uses ODBC and Scalable SQL (also referred to as the SQL Interface) to create and maintain dictionary and table definitions, your database is able to work with Scalable SQL, ODBC, and ODBC-based third-party tools.

DDF Ease provides the following capabilities:

- Support for Scalable SQL v3.x and Scalable SQL v4 DDF file formats.
- 100% compatible with ODBC and Scalable SQL.
- Create, open, and delete DDFs.
- Create new table definitions.
- Create table definitions for existing Btrieve data files.
- Drop table definitions.
- Alter table column and index definitions.
- Alter table location.
- Add/Drop table columns.
- Add/Drop named indexes.
- Display/print table data, table definitions, and statistics.
- Convert database dictionary formats.

- Check the database for inconsistencies.
- Enable/disable database security.

System Requirements

DDF Ease is installed as part of the default Pervasive.SQL Win32 Client installation. This utility requires that you have ODBC Interface 2.5 installed, which is also a part of the typical installation. However, if you choose to uninstall ODBC, you will not be able to run DDF Ease.

We recommend that you use the default Pervasive.SQL components included in the Pervasive.SQL installation program. However, you can use the Scalable SQL 3.01 engine, with the limitation that DDF Ease will not support adding and removing table columns and indexes. Other configurations, such as using a remote Btrieve 6.15 server engine with a Pervasive.SQL client, should work but have not been tested. The Btrieve file format of the DDF files must be 6.x or higher, but the table data files can be of pre-6.x Btrieve file formats.

Btrieve 7 engines are backward compatible with previous file format versions. To modify the Btrieve file version, use the Win32 Setup utility that is included with your Pervasive.SQL product and configure the following:

Microkernel Database Engine: File Settings: Create File Version: 7.x | 6.x

Choose 7.x as the setting if your database is of Btrieve 7.x file format, and choose 6.x if your Btrieve database is of 6.x format.

> **Note:** WARNING! Be aware that the "Create File Version" setting affects all applications running on the server. If your production applications require that your server is configured for 6.x and you need to create DDFs for 7.x, you will need to either (1) schedule a time to change the setting and restart your server for the DDF Ease work and then reset it when you've finished, or (2) install the appropriate software on a non-production server and do the work there so that there are no side affects from production applications and the Create File Version setting.

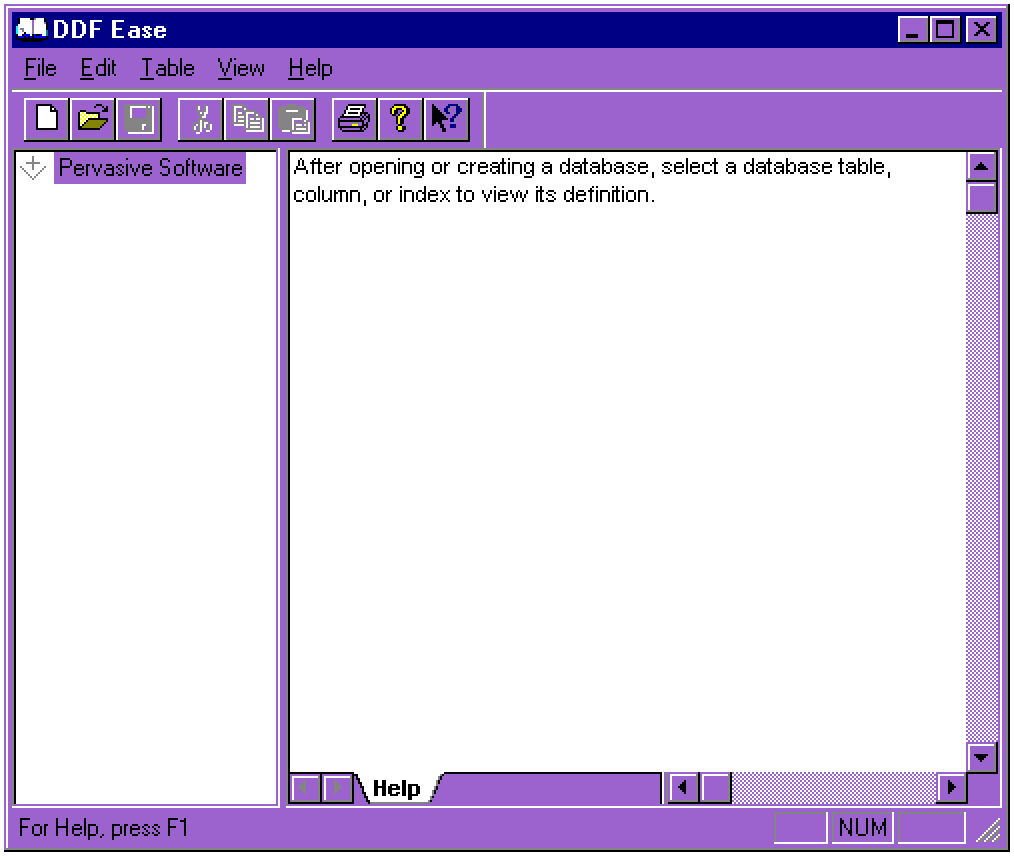
Starting DDF Ease

To start DDF Ease:

1. From the **Start** menu, select **Programs** and then select **Pervasive SQL 7** .

2. Select **DDF Ease (Win32)** to open the **DDF Ease** main window.

Figure 4-1
DDF Ease Main Window



After opening or creating a database, select a database table, column, or index to view its definition.

For Help, press F1

The following table lists the menus on the main window:

Table 4-1
DDF Ease Main Window Features

Menu	Contents
File	From the file menu, you can perform the following functions:#• Create a new database. • Open a database. • Close a database. • Delete a database.

- Save a database
- Check a database.
- Convert a data dictionary.
- Set database security.
- Print current table information.
- Print preview of current table information.
- Set Printer options.
- Exit DDF Ease.

Edit Allows you to perform Undo, Cut, Copy and Paste functions.

Table From this menu, you can create a table, drop a table, or show the system tables for the current database (X$File, X$Field, X$Index).

View This menu gives the option of viewing the toolbar and status bar.

Help You can access DDF Ease online Help or view information about DDF Ease. DDF Ease also incorporates context-sensitive help.

Creating DDFs for Existing Btrieve Files

This section takes you through the process of creating data dictionary files for existing Btrieve files. In this example, you are going to create the sample Patients database, copy Btrieve files into the database directory, and then create table definitions for each Btrieve file. The Patients database includes the following tables:

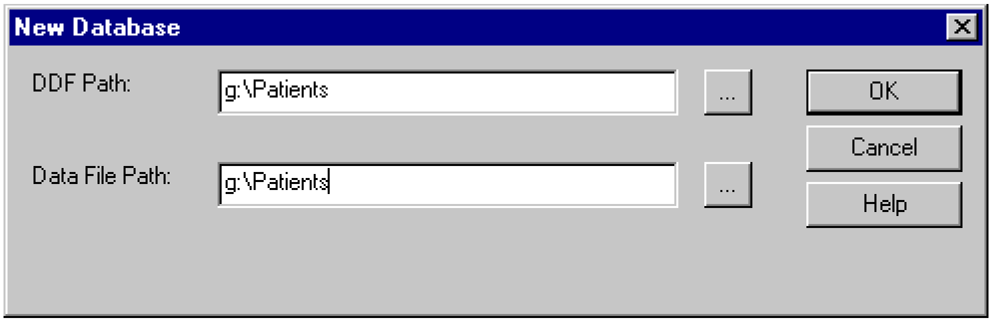
Table 4-2
Example Patients Database Tables

Patients	The patients who go to the same doctor's office.The Btrieve data file is patients.dta.
Appointments	Appointments for each patient. The Btrieve data file is patapp.dta.
Procedures	Procedures performed on each patient.The Btrieve data file is patproc.dta.

To create the Patients database:

1. Click 📂 on the toolbar or select **New Database** from the File menu. The New Database dialog box (Figure 4-2) is displayed.

Figure 4-2
New Database Dialog Box



DDF Path: g:\Patients

Data File Path: g:\Patients

2. If you are running DDF Ease from a Windows 95/NT client, enter the name of the remote directory where you want the database to reside in the DDF Path field and name it “Patients”.

   Although optional, you can specify the directory where you want the data files to reside in the Data File Path field.

3. Click **OK** to create the database. If the directory does not exist, you are prompted to create it. Click **OK** in the message pop-up dialog box.

> **Note:** The DDF Path is the directory where you want your data dictionary files (e.g., file.ddf, index.ddf, field.ddf) to reside. The Data File Path is the directory where your Btrieve data files (e.g., *.dta, *.mkd, etc.) reside.

4. Copy the following files from your c:\Pvsw\demodata\odbc directory (or the local directory that contains your Pervasive.SQL 7 software) to your Patients directory:

PATIENTS.DTA

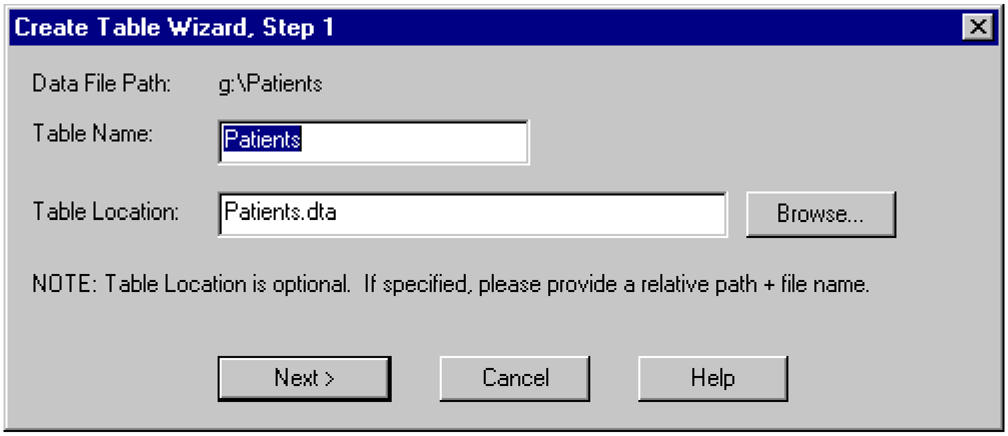
PATAPP.DTA

PATPROC.DTA

5. Create table definitions for each of these Btrieve files.

a. In the DDF Ease main window, click on the Patients database and then select **Create** from the **Table** menu. In the Create Table Wizard, Step 1 dialog box (Figure 4-3), enter “Patients” in the Table Name field and then use the **Browse** button to select the Patients.dta from the newly created Patients database directory. Click **Next** .

Figure 4-3
Create Table Wizard, Step 1 Dialog Box



Create Table Wizard, Step 1

Data File Path: g:\Patients

Table Name: Patients

Table Location: Patients.dta

NOTE: Table Location is optional. If specified, please provide a relative path + file name.

A message is displayed indicating that your Btrieve file is in v6 format and your database dictionary v7 format.

b. Click **Yes** to confirm this message.

The Create Table Wizard, Step 2 – Patients dialog box is displayed. Because you have created a table with an existing Btrieve file, DDF Ease can deduce some information about the data in the file. The dialog box displays index and column information.

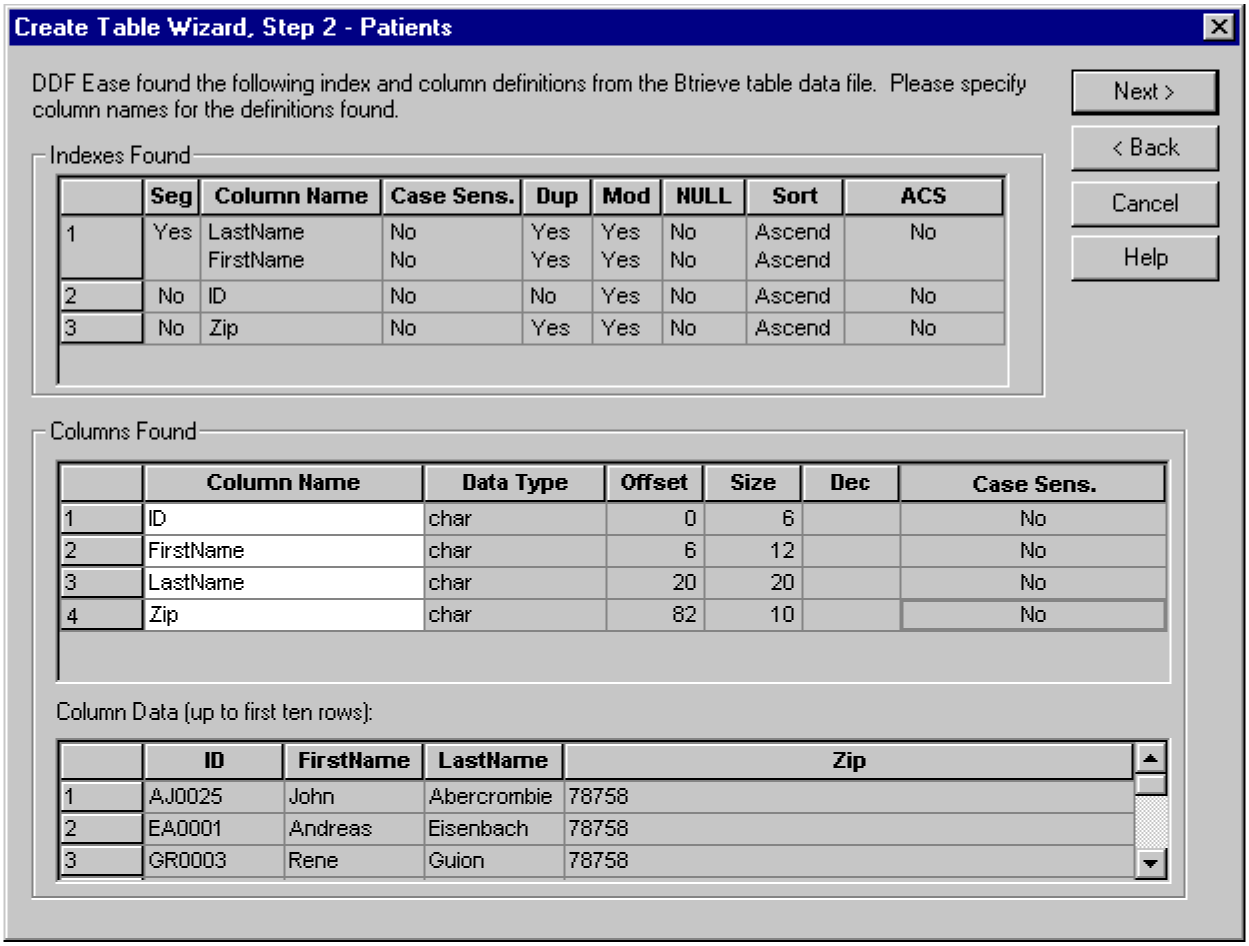
c. Specify the column names in the **Columns Found** section of the dialog box using the information in the following figure.

Notice that the newly entered column names are displayed in the **Indexes Found** and the **Column Data** sections.

d. Click **Next** when you have specified each column name.

Notice the new names appear in the **Column Data** section after you enter them. For more information about this dialog box, click **Help** or press F1.

Figure 4-4
Create Table Wizard, Step 2 - Patients Dialog Box



Create Table Wizard, Step 2 - Patients

DDF Ease found the following index and column definitions from the Btrieve table data file. Please specify column names for the definitions found.

Indexes Found

	Seg	Column Name	Case Sens.	Dup	Mod	NULL	Sort	ACS
1	Yes	LastName FirstName	No No	Yes Yes	Yes Yes	No No	Ascend Ascend	No
2	No	ID	No	No	Yes	No	Ascend	No
3	No	Zip	No	Yes	Yes	No	Ascend	No

Columns Found

	Column Name	Data Type	Offset	Size	Dec	Case Sens.
1	ID	char	0	6		No
2	FirstName	char	6	12		No
3	LastName	char	20	20		No
4	Zip	char	82	10		No

Column Data (up to first ten rows):

	ID	FirstName	LastName	Zip
1	AJ0025	John	Abercrombie	78758
2	EA0001	Andreas	Eisenbach	78758
3	GR0003	Rene	Guion	78758

e. The Create Table Wizard, Step 3 – Patients dialog box (Figure 4-6) is displayed.

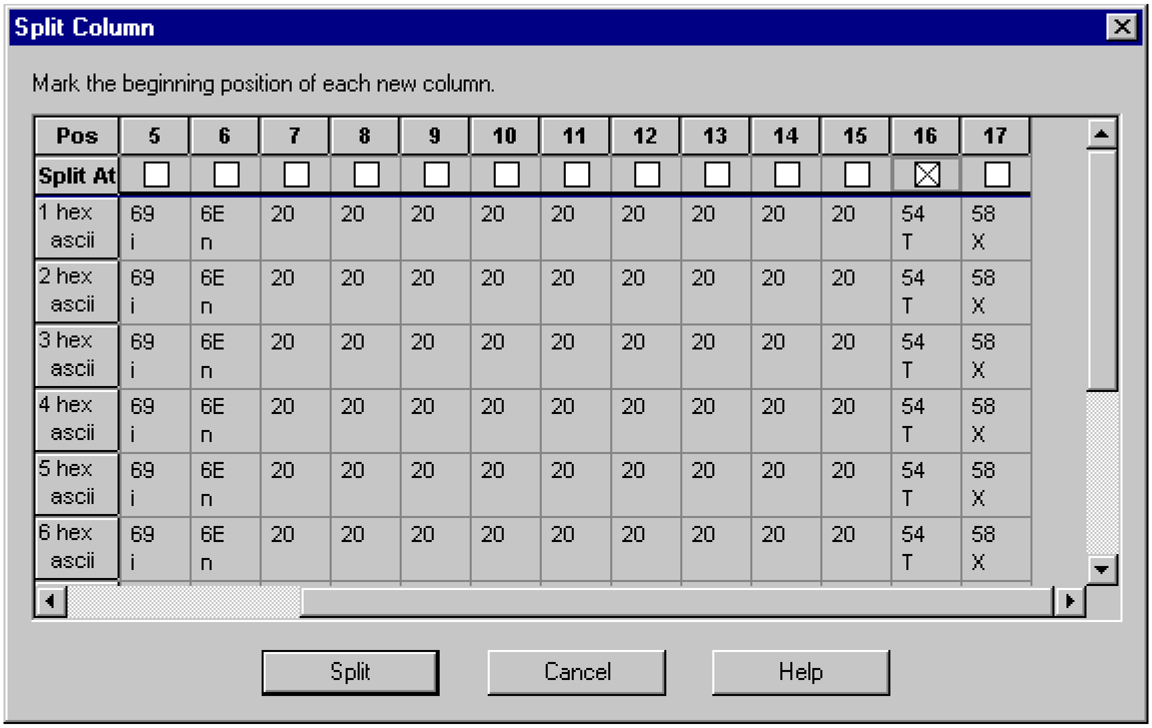
This dialog box directs you to specify column definitions for columns that have an “Unknown” data type. Because Btrieve data files are strings of bytes and some contain indexes, you may want to view the data to decide whether to split the column. You can view the data by clicking in a column name that has a data type that you can change and then clicking **Split Column** to actually see the data in hex and ASCII formats. When you view the data in the Split Column dialog box (Figure 4-5), you may notice the data is hard to recognize. For this example, the only column you will split is the third Unnamed column (which you will later name “City”).

In the Split Column dialog box, you will notice that the data looks like more address information: city and state.

f. Split the column into two sections by marking the appropriate break. For this example, you are going to mark the “T” in TX (position 16). Click **Split** when you are finished.

For more information about splitting columns, see “Helpful Hints for Splitting Columns” in the DDF Ease Online Help system.

Figure 4-5
Split Column Dialog Box



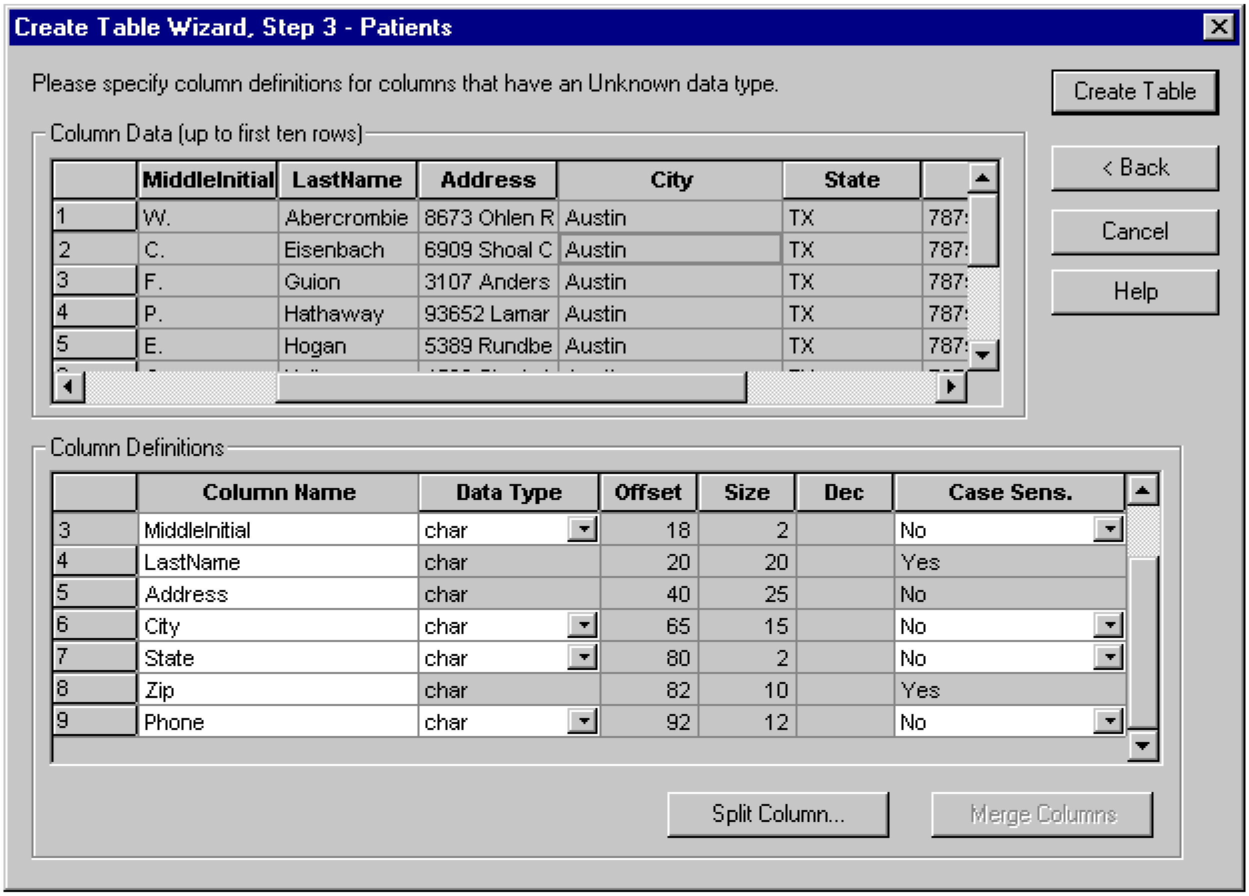
Split Column

Mark the beginning position of each new column.

Pos	5	6	7	8	9	10	11	12	13	14	15	16	17
Split At	□	□	□	□	□	□	□	□	□	□	□	☒	□
1 hex ascii	69 i	6E n	20	20	20	20	20	20	20	20	20	54 T	58 X
2 hex ascii	69 i	6E n	20	20	20	20	20	20	20	20	20	54 T	58 X
3 hex ascii	69 i	6E n	20	20	20	20	20	20	20	20	20	54 T	58 X
4 hex ascii	69 i	6E n	20	20	20	20	20	20	20	20	20	54 T	58 X
5 hex ascii	69 i	6E n	20	20	20	20	20	20	20	20	20	54 T	58 X
6 hex ascii	69 i	6E n	20	20	20	20	20	20	20	20	20	54 T	58 X

g. Use the information in Figure 4-6 to complete the column definitions.

Figure 4-6
Create Table Wizard, Step 3 - Patients Dialog Box



Create Table Wizard, Step 3 - Patients

Please specify column definitions for columns that have an Unknown data type.

Column Data (up to first ten rows)

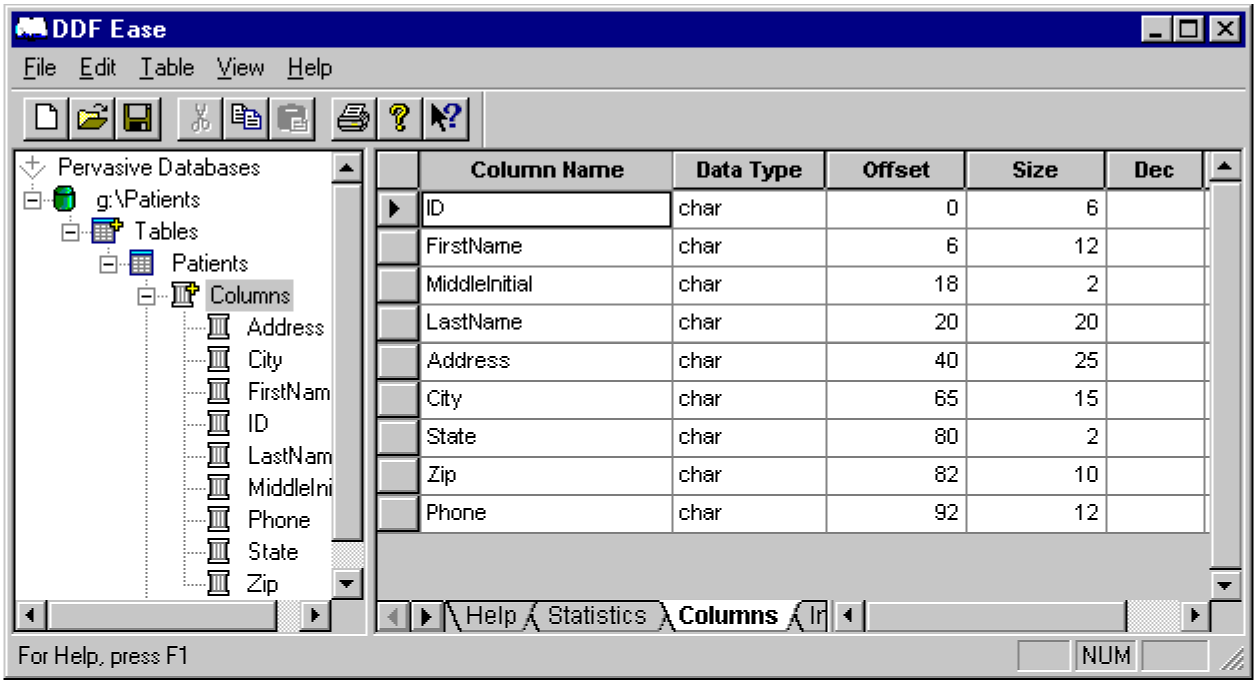
	MiddleInitial	LastName	Address	City	State	
1	W.	Abercrombie	8673 Ohlen R	Austin	TX	787:
2	C.	Eisenbach	6909 Shoal C	Austin	TX	787:
3	F.	Guion	3107 Anders	Austin	TX	787:
4	P.	Hathaway	93652 Lamar	Austin	TX	787:
5	E.	Hogan	5389 Rundbe	Austin	TX	787:

Column Definitions

	Column Name	Data Type	Offset	Size	Dec	Case Sens.
3	MiddleInitial	char	18	2		No
4	LastName	char	20	20		Yes
5	Address	char	40	25		No
6	City	char	65	15		No
7	State	char	80	2		No
8	Zip	char	82	10		Yes
9	Phone	char	92	12		No

h. When you have finished naming the columns and specifying data types, click **Create Table** . The new table is displayed in the directory tree of the main window. The default view displays the **Statistics** tab. Click the **Columns** tab to view the table structure as illustrated in Figure 4-7. For more information about data types supported in Pervasive.SQL 7 refer to *SQL Language Reference* or the DDF Ease online help system (press **F1** or click **Help** from any dialog box).

Figure 4-7
DDF Ease Main Window - Patients Database Example

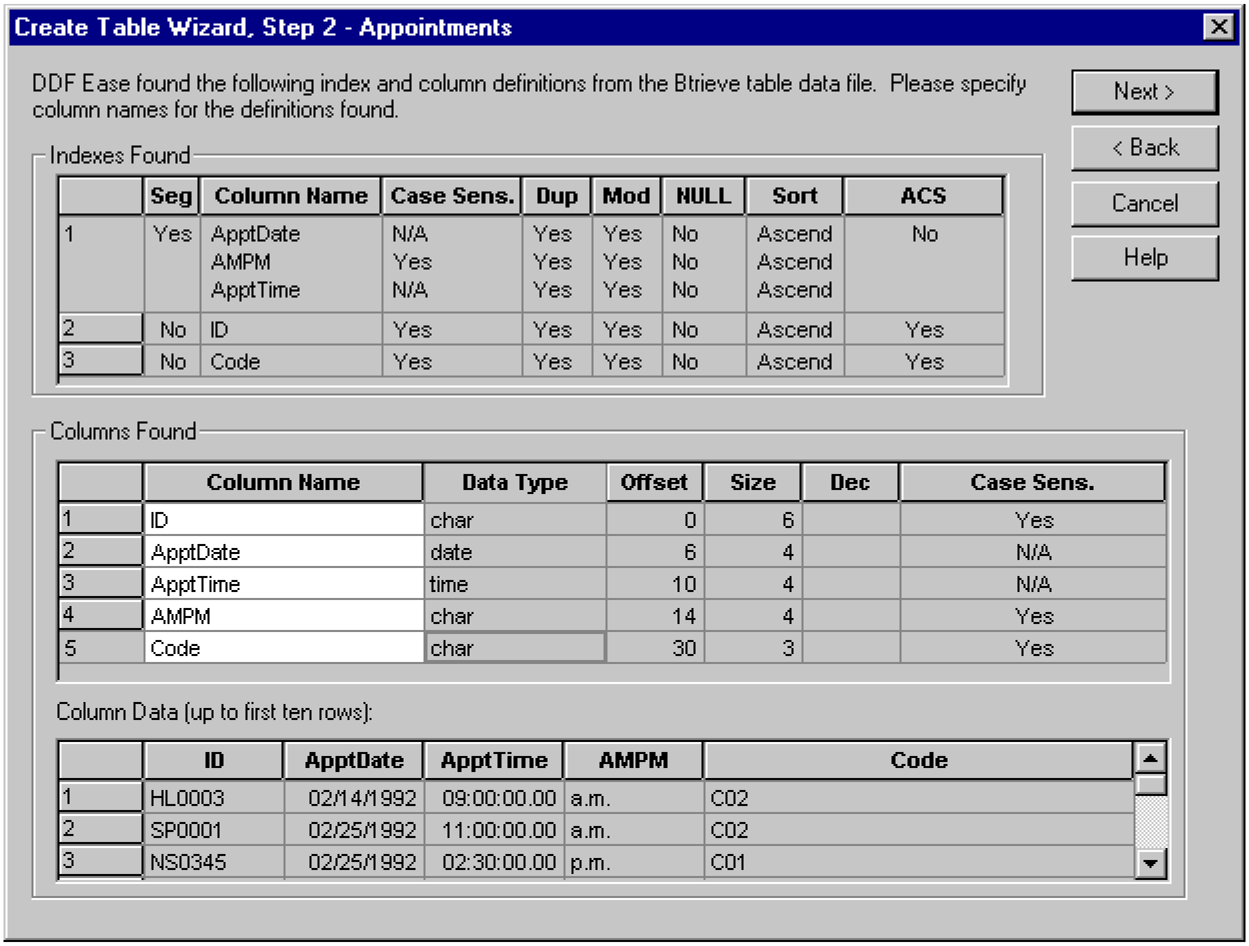


Column Name	Data Type	Offset	Size	Dec
ID	char	0	6	
FirstName	char	6	12	
MiddleInitial	char	18	2	
LastName	char	20	20	
Address	char	40	25	
City	char	65	15	
State	char	80	2	
Zip	char	82	10	
Phone	char	92	12	

For Help, press F1

6. Repeat the procedures in the previous step to create tables for Appointments using the PATAPP.DTA file. In the Create Table Wizard, Step 2 dialog box, name the columns as illustrated in Figure 4-8.

Figure 4-8
Create Table Wizard, Step 2 - Appointments Dialog Box



DDF Ease found the following index and column definitions from the Btrieve table data file. Please specify column names for the definitions found.

Indexes Found

	Seg	Column Name	Case Sens.	Dup	Mod	NULL	Sort	ACS
1	Yes	ApptDate AMPM ApptTime	N/A Yes N/A	Yes Yes Yes	Yes Yes Yes	No No No	Ascend Ascend Ascend	No
2	No	ID	Yes	Yes	Yes	No	Ascend	Yes
3	No	Code	Yes	Yes	Yes	No	Ascend	Yes

Columns Found

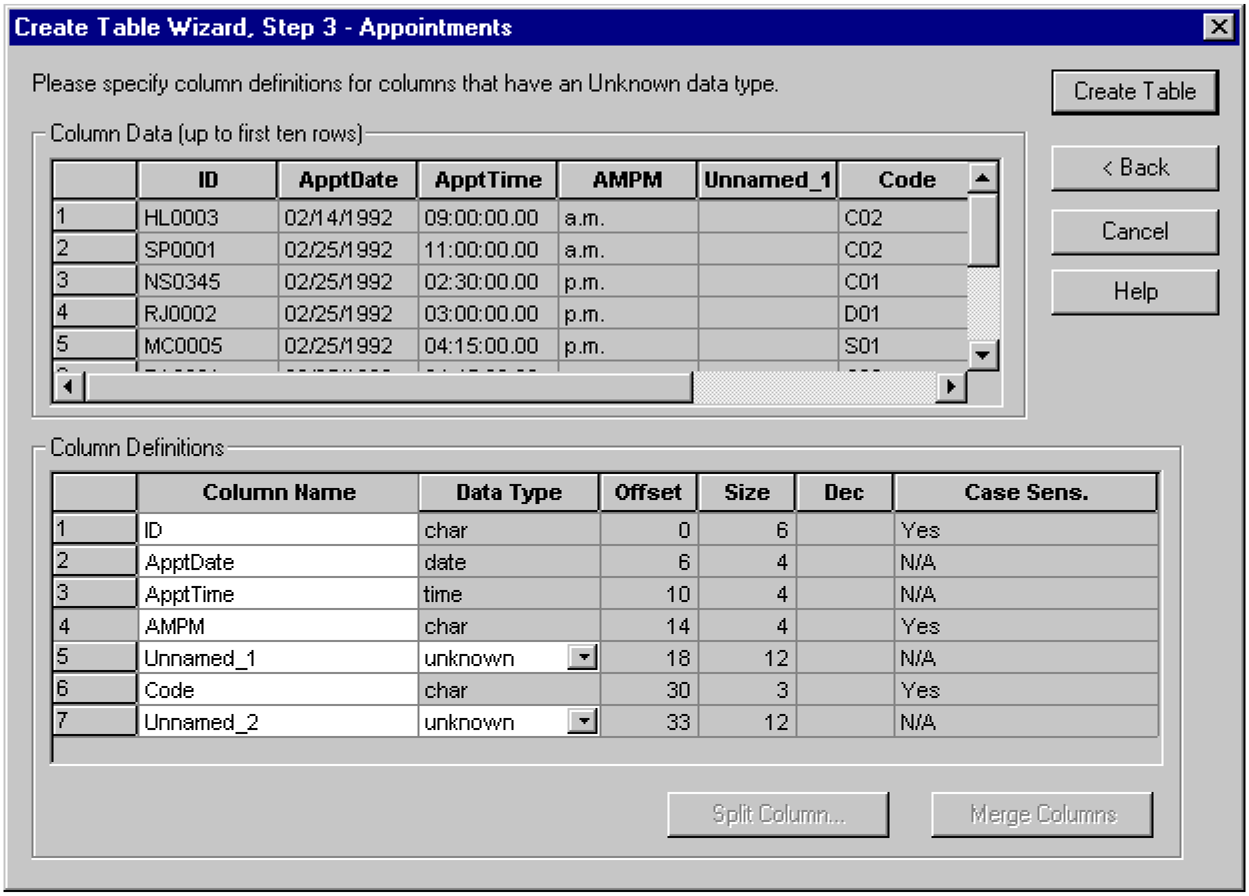
	Column Name	Data Type	Offset	Size	Dec	Case Sens.
1	ID	char	0	6		Yes
2	ApptDate	date	6	4		N/A
3	ApptTime	time	10	4		N/A
4	AMPM	char	14	4		Yes
5	Code	char	30	3		Yes

Column Data (up to first ten rows):

	ID	ApptDate	ApptTime	AMPM	Code
1	HL0003	02/14/1992	09:00:00.00	a.m.	C02
2	SP0001	02/25/1992	11:00:00.00	a.m.	C02
3	NS0345	02/25/1992	02:30:00.00	p.m.	C01

a. Click **Next** . The Create Table Wizard, Step 3 dialog box is displayed.

Figure 4-9
Create Table Wizard, Step 2 - Appointments Dialog Box



Create Table Wizard, Step 3 - Appointments

Please specify column definitions for columns that have an Unknown data type.

Column Data (up to first ten rows)

	ID	ApptDate	ApptTime	AMPM	Unnamed_1	Code
1	HL0003	02/14/1992	09:00:00.00	a.m.		C02
2	SP0001	02/25/1992	11:00:00.00	a.m.		C02
3	NS0345	02/25/1992	02:30:00.00	p.m.		C01
4	RJ0002	02/25/1992	03:00:00.00	p.m.		D01
5	MC0005	02/25/1992	04:15:00.00	p.m.		S01

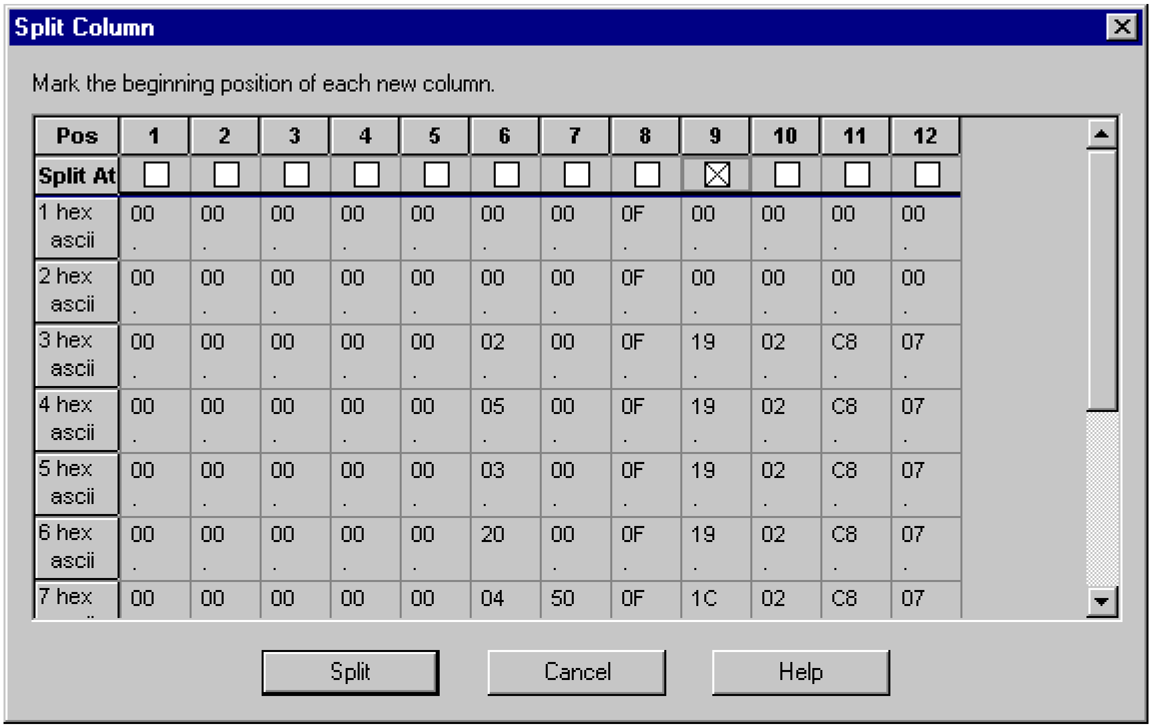
Column Definitions

	Column Name	Data Type	Offset	Size	Dec	Case Sens.
1	ID	char	0	6		Yes
2	ApptDate	date	6	4		N/A
3	ApptTime	time	10	4		N/A
4	AMPM	char	14	4		Yes
5	Unnamed_1	unknown	18	12		N/A
6	Code	char	30	3		Yes
7	Unnamed_2	unknown	33	12		N/A

b. Notice that there are only two columns that are labeled "Unnamed" (columns 5 and 7). Before you can specify a data type, you may need to look at the data in the column. For column 5, click on its row and then select **Split Column** .

You will see that the data contains last names, which means this is a Char data type. Click **Cancel** on the Split Column dialog box, name the column "Doctor" and select char from the drop-down list. For column 7, look at the data in the Split Column dialog box. There is an obvious pattern of HEX 0F for all data at position 8. For the Appointments table, you will split the seventh column at the position 9 as illustrated in Figure 4-10.

Figure 4-10
Split Column Dialog Box



Split Column

Mark the beginning position of each new column.

Pos	1	2	3	4	5	6	7	8	9	10	11	12
Split At	□	□	□	□	□	□	□	□	☒	□	□	□
1 hex	00	00	00	00	00	00	00	0F	00	00	00	00
ascii	.	.	.	.	.	.	.	.	.	.	.	.
2 hex	00	00	00	00	00	00	00	0F	00	00	00	00
ascii	.	.	.	.	.	.	.	.	.	.	.	.
3 hex	00	00	00	00	00	02	00	0F	19	02	C8	07
ascii	.	.	.	.	.	.	.	.	.	.	.	.
4 hex	00	00	00	00	00	05	00	0F	19	02	C8	07
ascii	.	.	.	.	.	.	.	.	.	.	.	.
5 hex	00	00	00	00	00	03	00	0F	19	02	C8	07
ascii	.	.	.	.	.	.	.	.	.	.	.	.
6 hex	00	00	00	00	00	20	00	0F	19	02	C8	07
ascii	.	.	.	.	.	.	.	.	.	.	.	.
7 hex	00	00	00	00	00	04	50	0F	1C	02	C8	07

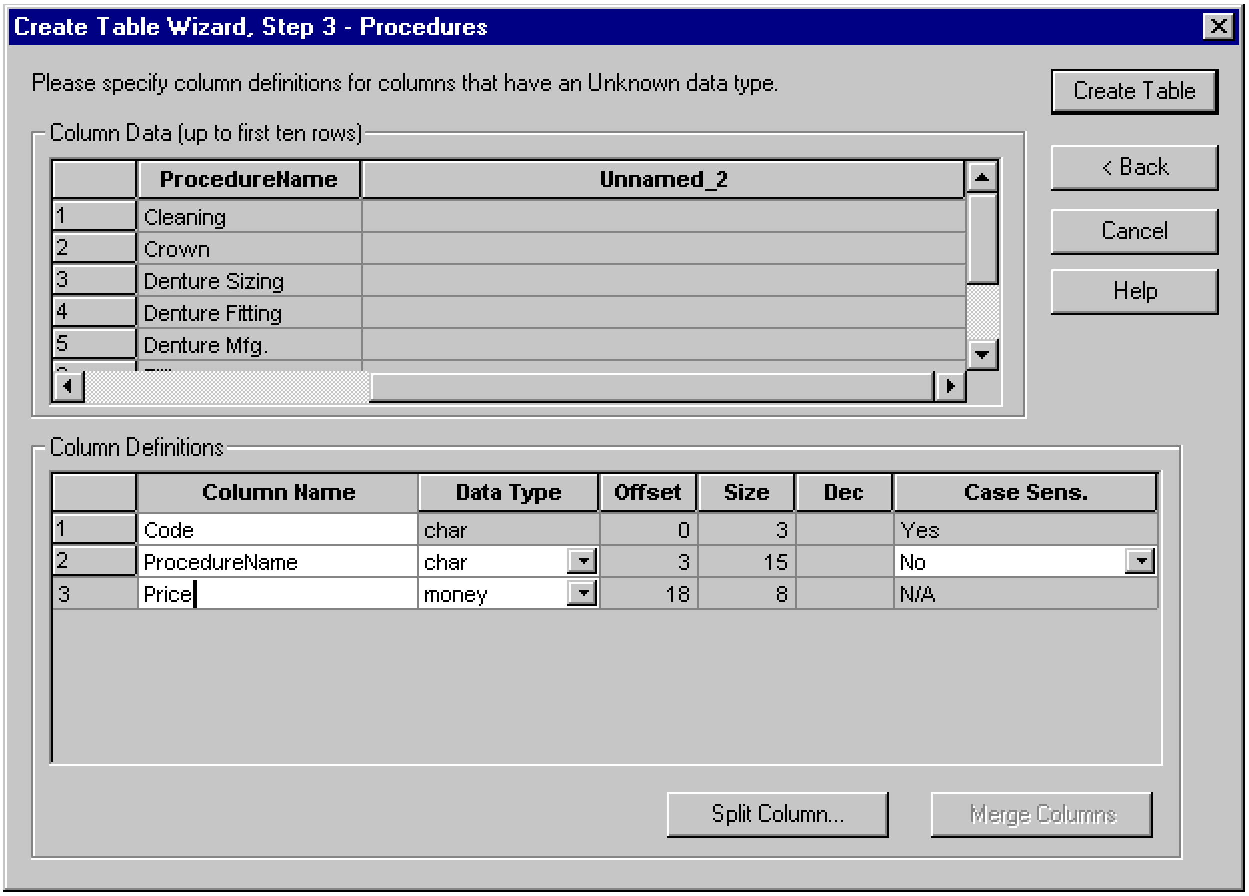
c. Name column 7 AmtPaid and specify it as money data type. For column 8, click **Split Column** to view the data. Because it is a four byte data type and the Hex values show a variation of numbers, it is a good candidate for a Date data type. Click **Cancel** , name this column “DatePaid”, and specify it as a Date data type.   Click **Create Table**   to create the Appointments table.

7. Repeat the previous steps for the Procedures table, using the PATPROC.DTA file. In the Create Table Wizard, Step 2 dialog box, name the column “Code” and then click **Next** .

a. In the Create Table Wizard, Step 3 dialog box, click in the Unnamed_1 column and then click **Split Column** to view the data so that you can decide whether to split the column. When you view the data in the Split Column dialog box, you will notice the first 14 bytes of data are easy to recognize words (in this case, names of procedures). Split the column at position 16 as this is the start of a new character set and then click **Split Column** .

Notice there are now two columns, Unnamed_1 and Unnamed_2. Name Unnamed_1 “ProcedureName” and define it as a char data type. Because Unnamed_2 contains characters that are not printable as ASCII, it is not Numeric, Numericsa, or Numericsts; however, in this example, it is money. Ensure the data in dialog box reflects the information illustrated in Figure 4-11.

Figure 4-11
Create Table Wizard, Step 3 - Procedures Dialog Box



Create Table Wizard, Step 3 - Procedures

Please specify column definitions for columns that have an Unknown data type.

Column Data (up to first ten rows)

	ProcedureName	Unnamed_2
1	Cleaning	
2	Crown	
3	Denture Sizing	
4	Denture Fitting	
5	Denture Mfg.	

Column Definitions

	Column Name	Data Type	Offset	Size	Dec	Case Sens.
1	Code	char	0	3		Yes
2	ProcedureName	char	3	15		No
3	Price	money	18	8		N/A

b. Click **Create Table** .

8. Once you have created these three tables, you can display different views of the data. If you select Tables in the database directory tree, you see a summary of the three tables you created as indicated in Figure 4-12.

Figure 4-12
DDF Ease Main Window - Patients Database Example

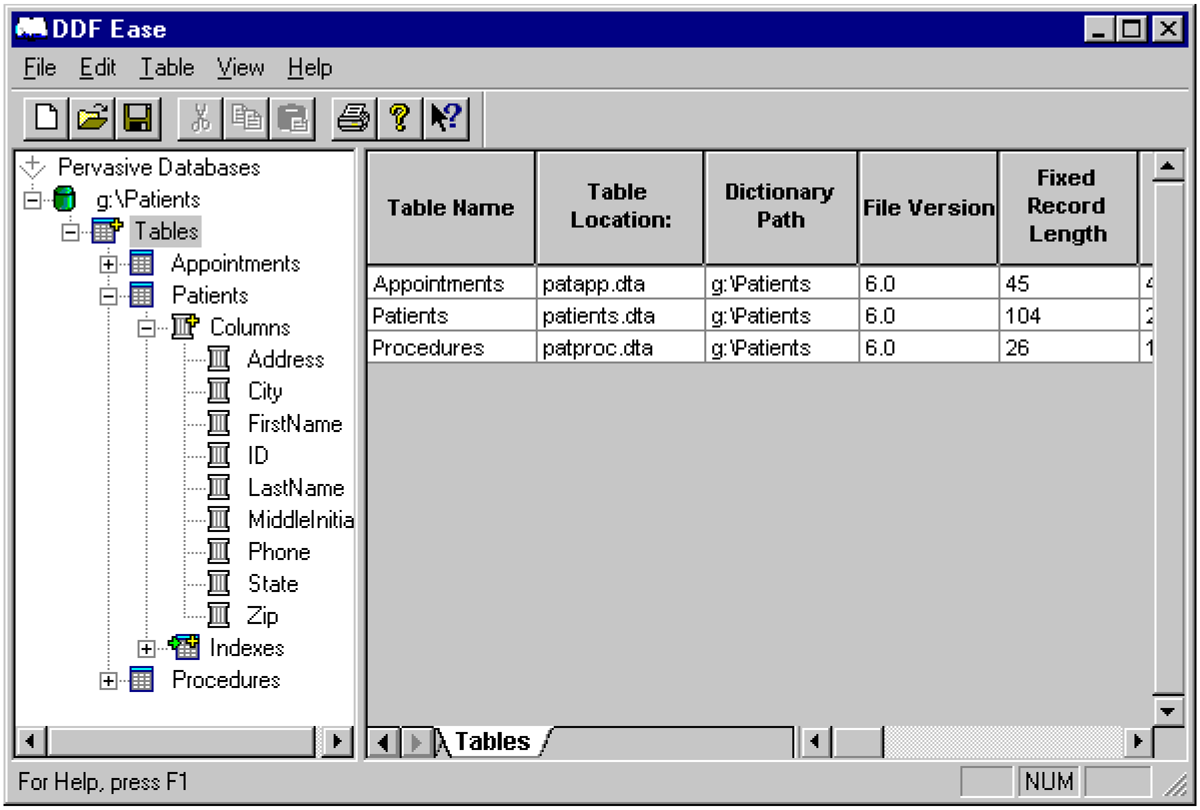


Table Name	Table Location:	Dictionary Path	File Version	Fixed Record Length
Appointments	patapp.dta	g:\Patients	6.0	45
Patients	patients.dta	g:\Patients	6.0	104
Procedures	patproc.dta	g:\Patients	6.0	26

Use the horizontal scroll bar to view more information. Click on a specific table name and then select a Tab (Statistics, Columns, Indexes, Data) to display more information.

Creating a Relational Database

This section gives you specific instructions for creating a simple relational database (using the SQL Interface [formerly Scalable SQL 4.x]) that keeps track of customer orders for a small company. The Customer Order database has the following tables:

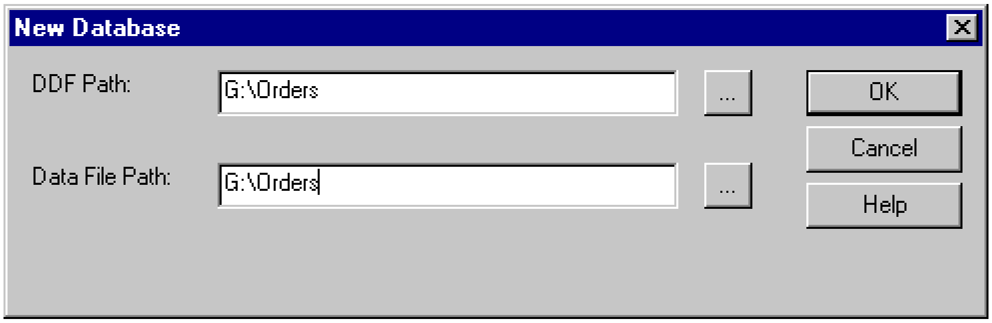
Customers	The customers that purchase the company's products.
Orders	Orders placed by customers.
SalesRep	The sales representatives that take customer orders.
SalesOffices	The sales office where the sales representative works.

To create a customer order database:

1. Create the Order Entry Database.

   a. To create the database, select **New Database** from the File menu, or you click the new file icon on the toolbar. The New Database dialog box is displayed.

Figure 4-13
DDF Ease New Database Dialog Box



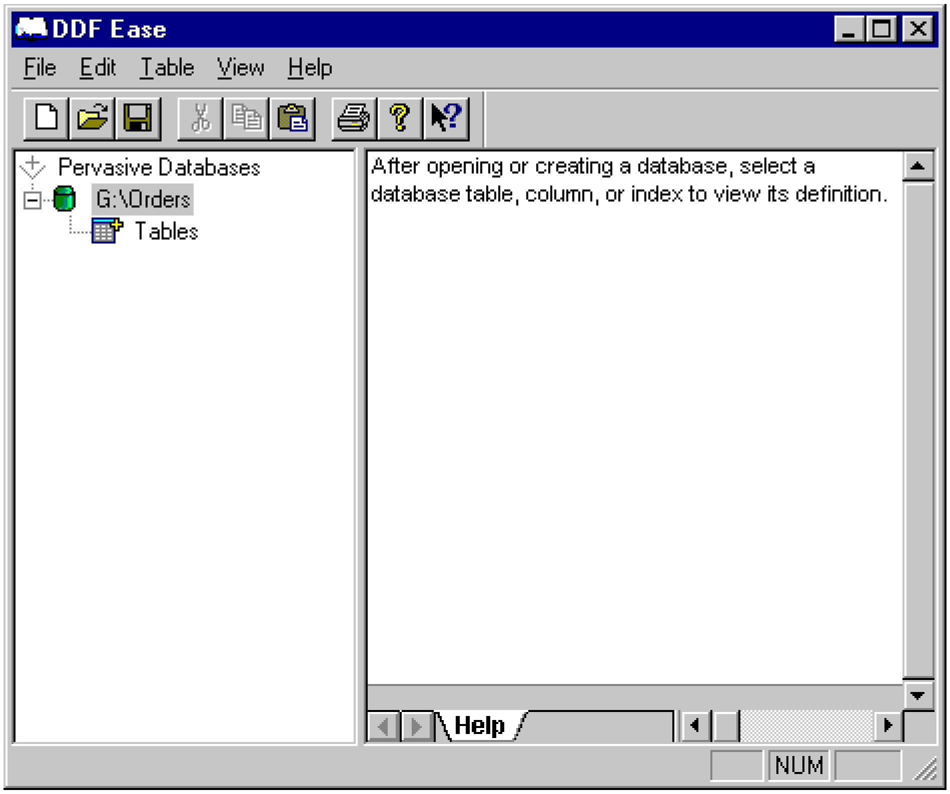
New Database

DDF Path: G:\Orders

Data File Path: G:\Orders

   b. You now have an empty database shown in the main tree view, as indicated in Figure 4-14. Next define the Customers, Orders, SalesReps, and SalesOffices tables.

Figure 4-14
.DDF Ease Main Window – Customer Orders Example



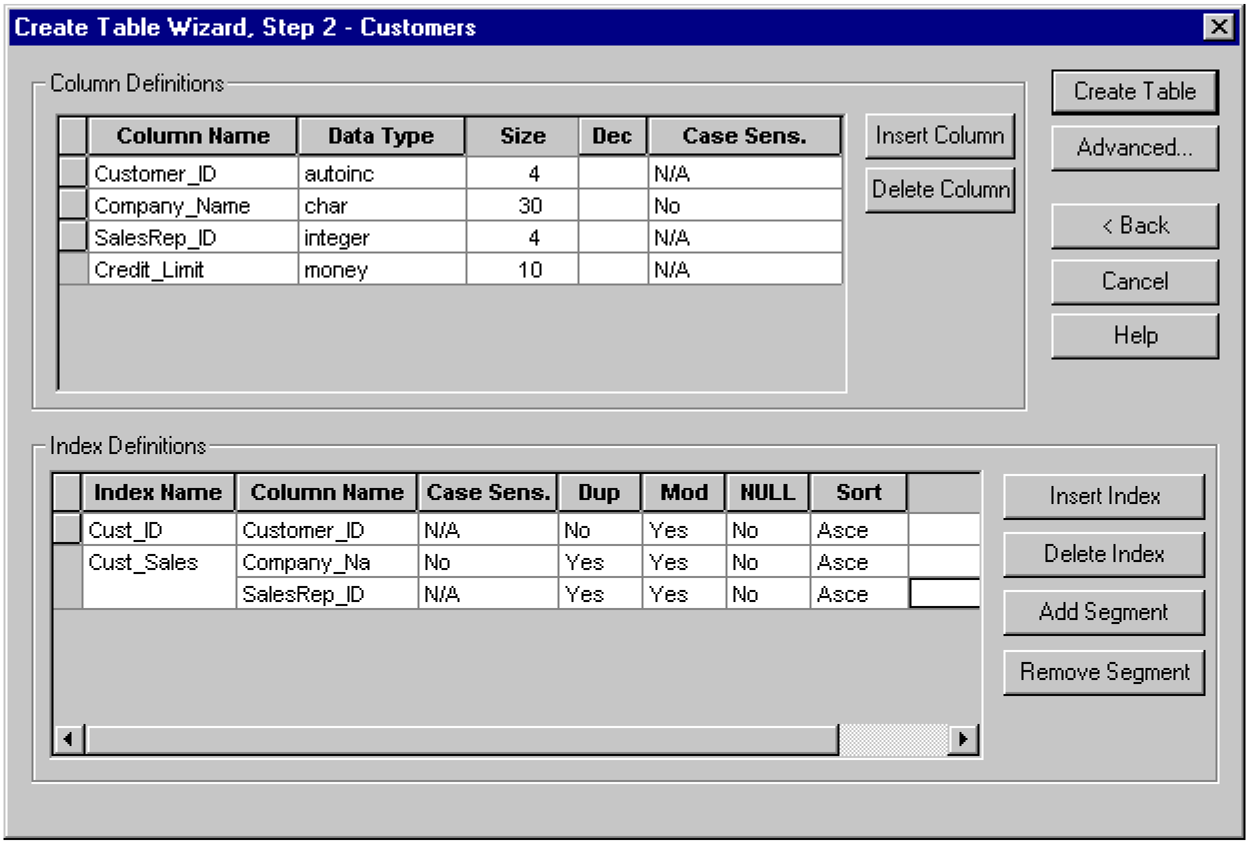
After opening or creating a database, select a database table, column, or index to view its definition.

2. Create the Customers Table.

   a. To create the Customers table, select **Create** from the Table menu. Enter the table name "Customers" and then click **Next** .

   b. Enter the columns and indexes as shown in Figure 4-15.

Figure 4-15
Create Table Wizard, Step 2 – Customers



Column Name	Data Type	Size	Dec	Case Sens.
Customer_ID	autoinc	4		N/A
Company_Name	char	30		No
SalesRep_ID	integer	4		N/A
Credit_Limit	money	10		N/A

Index Name	Column Name	Case Sens.	Dup	Mod	NULL	Sort
Cust_ID	Customer_ID	N/A	No	Yes	No	Asce
Cust_Sales	Company_Na	No	Yes	Yes	No	Asce
	SalesRep_ID	N/A	Yes	Yes	No	Asce

To enter a column name, double-click the default “Unnamed” so that it is highlighted and then type the new name. To insert another column below the first column, press Tab or the down-arrow key. Repeat these steps for inserting indexes.

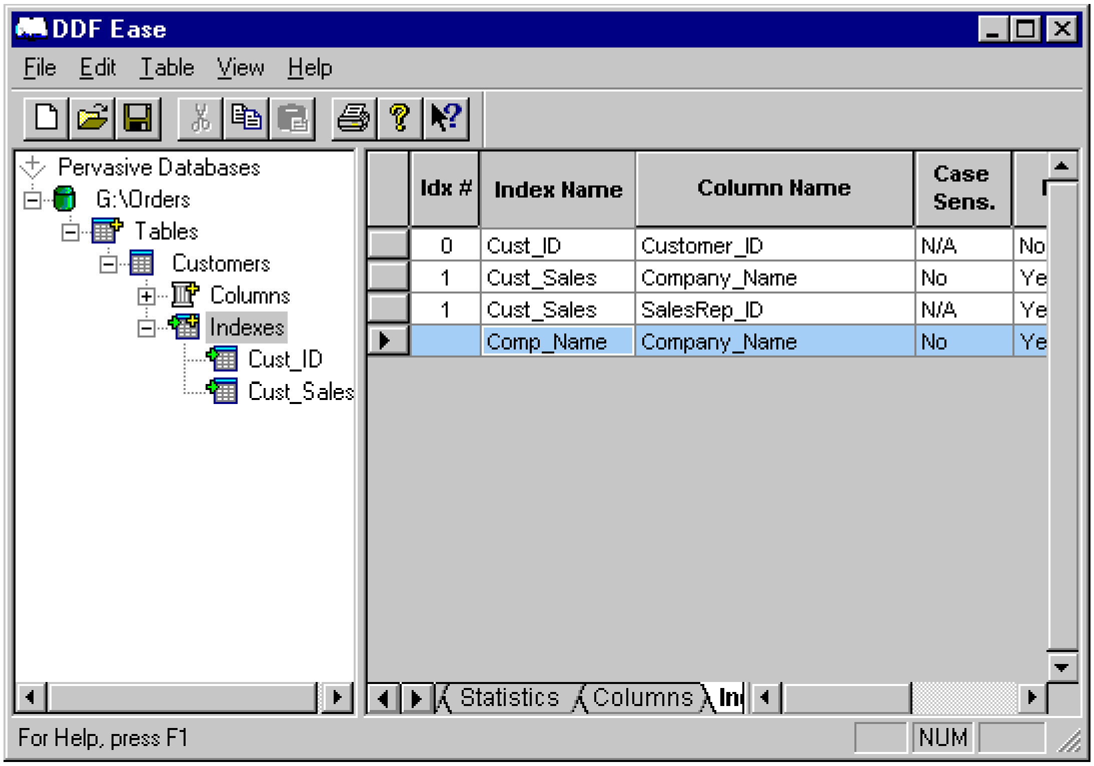
To insert a segmented index, first insert a new index and then click **Add Segment** . Scalable SQL 4 allows you to create segmented indexes with Autoinc. However, this is not supported in Scalable SQL 3.01. Also, remember that you cannot have the same index name as the column name; these names must be unique.

For specific information about completing column and index information, click **Help** on the Create Table Wizard, Step 2 dialog box. Click **Create Table** to create the table.

c. Suppose that later you want to add an index on Company_Name. In the main tree view, select the Customers table and then the **Indexes** tab. Click in the last row of the Indexes view and press the Down Arrow.

Notice in Figure 4-16 the row is highlighted in blue until you save the changes. Enter the index name “Comp_Name” and select the column “Company_Name”. Click **Save** from the toolbar.

Figure 4-16
DDF Ease Main Window – Insert Index Example

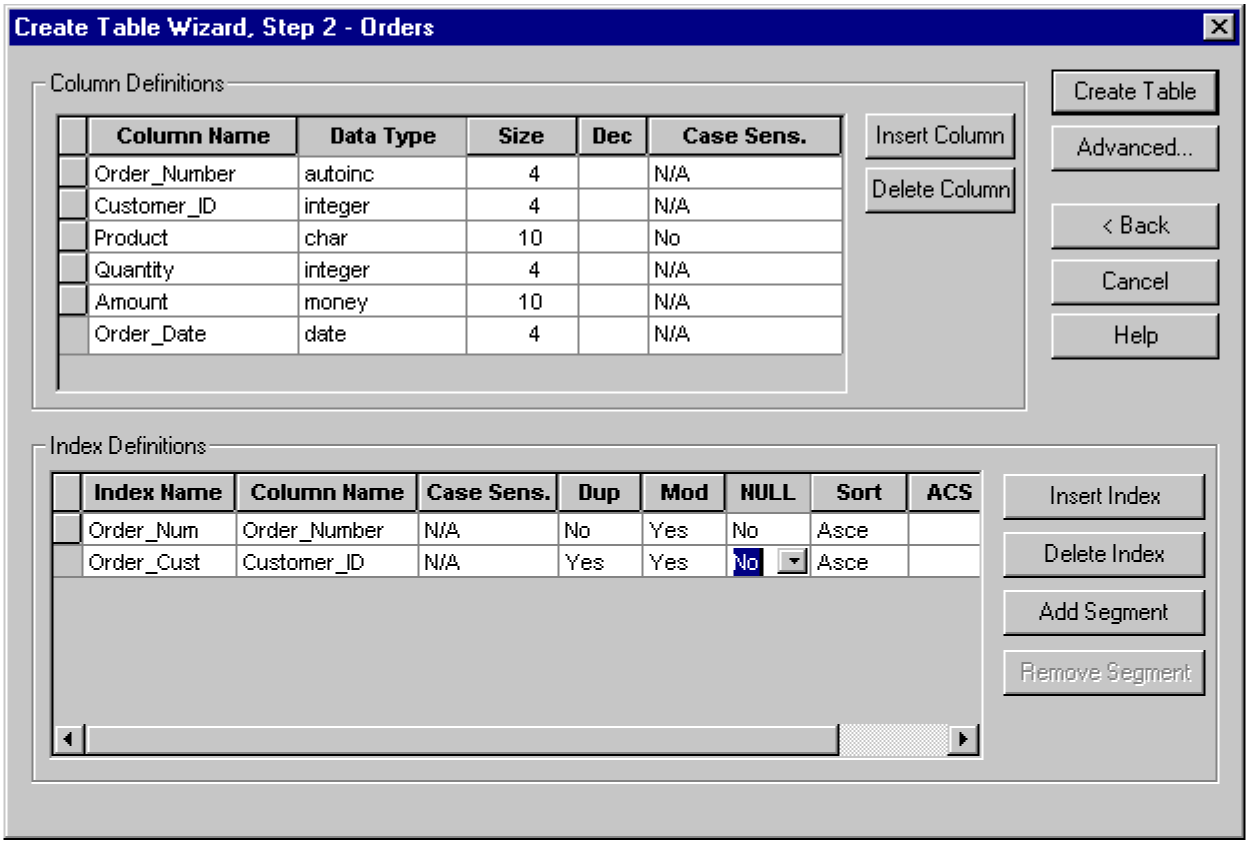


	Idx #	Index Name	Column Name	Case Sens.	
	0	Cust_ID	Customer_ID	N/A	No
	1	Cust_Sales	Company_Name	No	Ye
	1	Cust_Sales	SalesRep_ID	N/A	Ye
▶		Comp_Name	Company_Name	No	Ye

3. Create the Orders Table.

To create the Orders table, select **Create** from the Table menu. Enter “Orders” as the table name and click **Next** . Then enter column and index information as shown in Figure 4-17, and click **Create Table** . (For help, press F1 or the **Help** button on this dialog box.)

Figure 4-17
Create Table Wizard, Step 2 – Orders



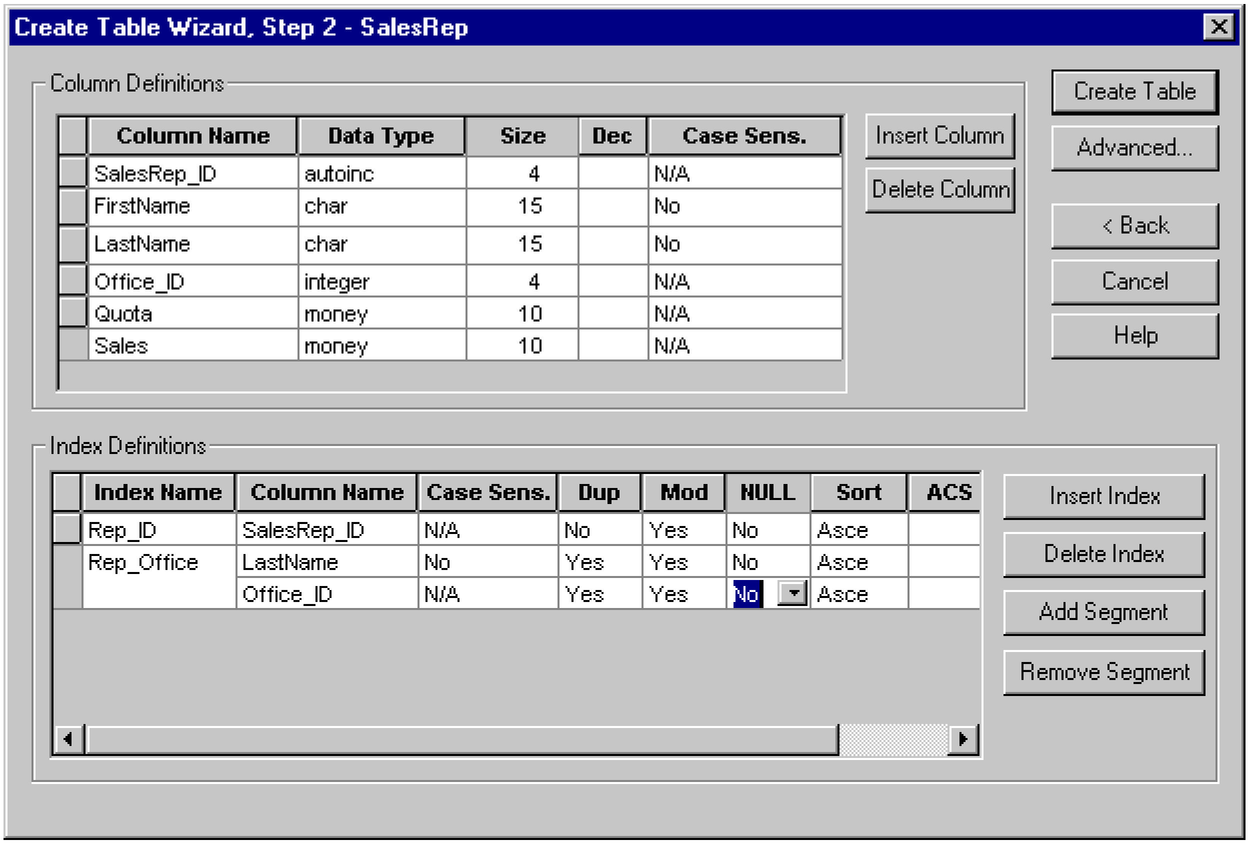
Column Name	Data Type	Size	Dec	Case Sens.
Order_Number	autoinc	4		N/A
Customer_ID	integer	4		N/A
Product	char	10		No
Quantity	integer	4		N/A
Amount	money	10		N/A
Order_Date	date	4		N/A

Index Name	Column Name	Case Sens.	Dup	Mod	NULL	Sort	ACS
Order_Num	Order_Number	N/A	No	Yes	No	Asce	
Order_Cust	Customer_ID	N/A	Yes	Yes	No	Asce	

4. Create the SalesRep Table.

To create the SalesRep table, select **Create** from the Table menu. Enter the following information as shown in Figure 4-18:

Figure 4-18
Create Table Wizard, Step 2 – SalesRep



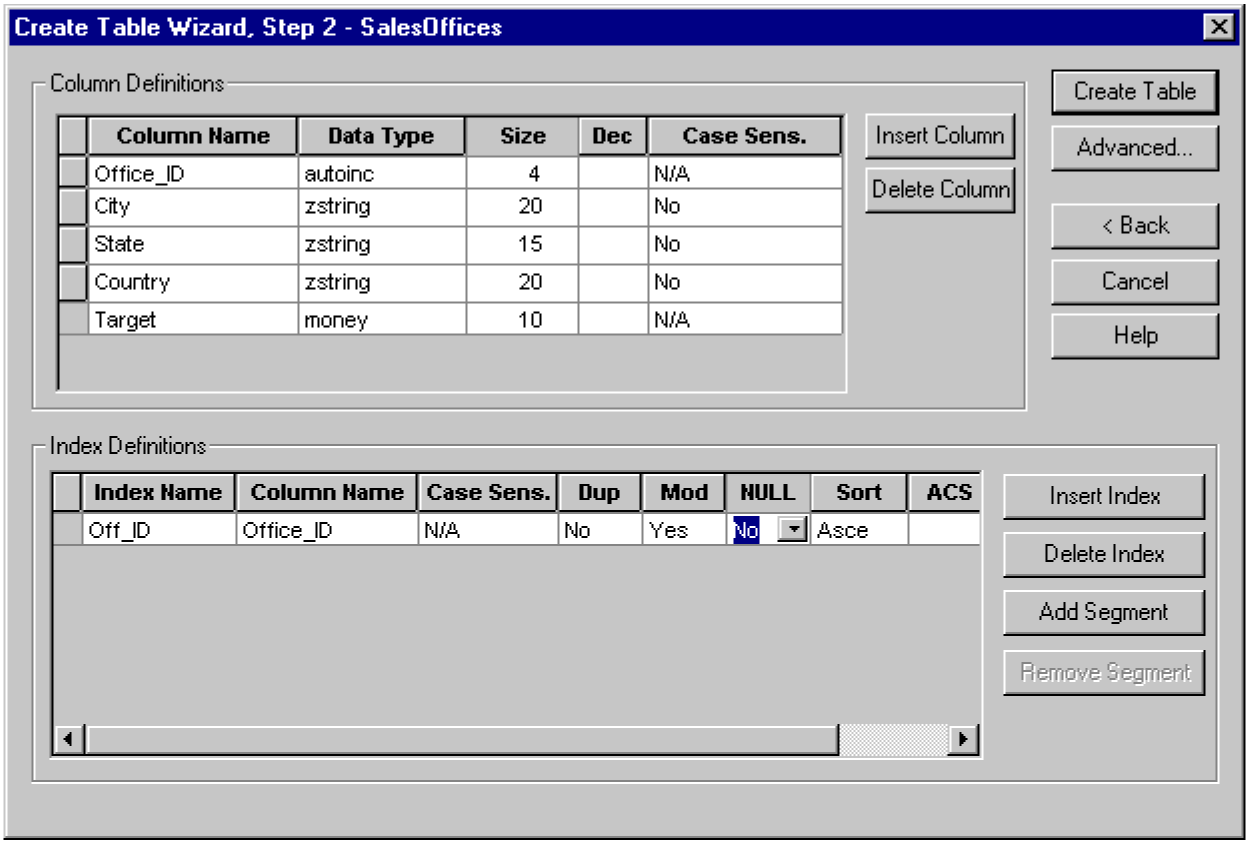
Column Name	Data Type	Size	Dec	Case Sens.
SalesRep_ID	autoinc	4		N/A
FirstName	char	15		No
LastName	char	15		No
Office_ID	integer	4		N/A
Quota	money	10		N/A
Sales	money	10		N/A

Index Name	Column Name	Case Sens.	Dup	Mod	NULL	Sort	ACS
Rep_ID	SalesRep_ID	N/A	No	Yes	No	Asce	
Rep_Office	LastName	No	Yes	Yes	No	Asce	
	Office_ID	N/A	Yes	Yes	No	Asce	

5. Create the SalesOffices Table.

d. To create the SalesOffices table, select **Create** from the Table menu. Enter the following information as shown in Figure 4-19.

Figure 4-19
Create Table Wizard, Step 2 – SalesOffices



Create Table Wizard, Step 2 - SalesOffices

Column Definitions

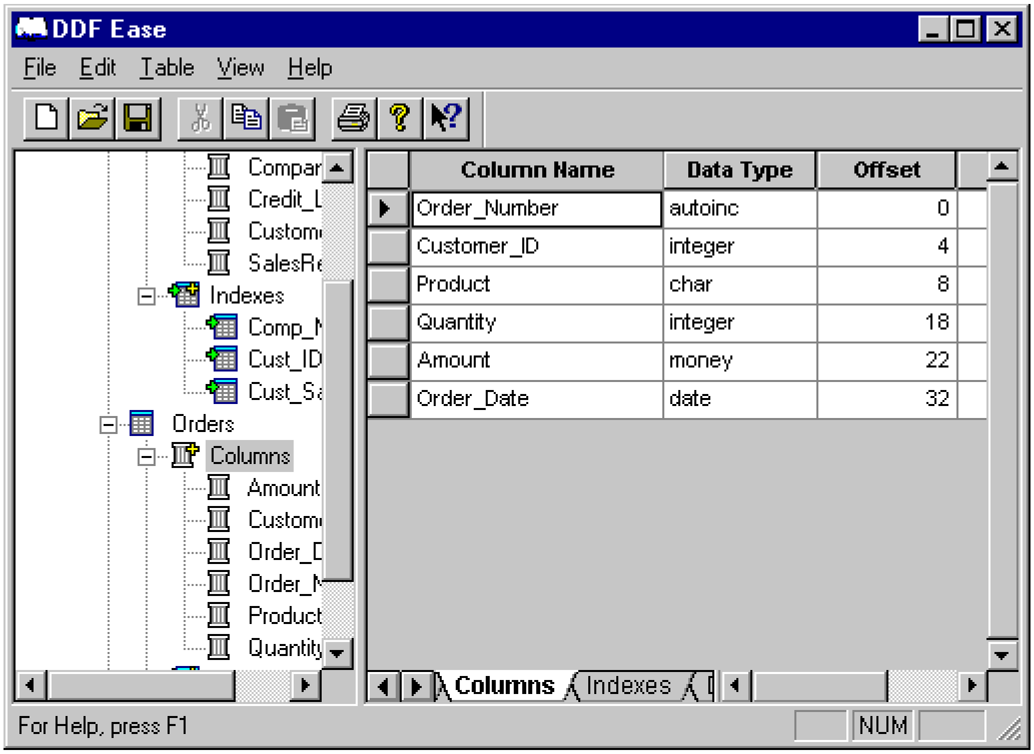
Column Name	Data Type	Size	Dec	Case Sens.
Office_ID	autoinc	4		N/A
City	zstring	20		No
State	zstring	15		No
Country	zstring	20		No
Target	money	10		N/A

Index Definitions

Index Name	Column Name	Case Sens.	Dup	Mod	NULL	Sort	ACS
Off_ID	Office_ID	N/A	No	Yes	No	Asce	

e. The following screen shows the tree view after adding tables. Some of the subfolders have been expanded to display columns and indexes. Notice how some of the information is not visible in this window. Use the horizontal scroll bar to the right of the tabs to view more information.

Figure 4-20
DDF Ease Main Window – Order Entry Database Example



Column Name	Data Type	Offset
Order_Number	autoinc	0
Customer_ID	integer	4
Product	char	8
Quantity	integer	18
Amount	money	22
Order_Date	date	32

6. Add Data to the Database.

At this time, DDF Ease does not support adding data to the database. However, you can execute statements in Pervasive's SQLScope utility to add new data to the database. (Refer to Chapter 9, "Executing SQL Statements with SQLScope," for more information.)

For example, suppose you want to add data for the sales offices. Here are the INSERT statements to add the sales office to the database:

```
INSERT INTO SalesOffices (City, State, Country, Target)
VALUES ('Austin', 'TX', 'USA', 800000)
INSERT INTO SalesOffices (City, State, Country, Target)
VALUES ('Boston', 'MA', 'USA', 510000)
INSERT INTO SalesOffices (City, State, Country, Target)
VALUES ('San Francisco', 'CA', 'USA', 500000)
```

Here is an INSERT statement to add the sales representative Andy Woodrif to the Austin sales office.

```
INSERT INTO SalesRep (FirstName, LastName, Office_ID, Quota, Sales)
VALUES ('Andy', 'Woodrif', 1, 250000, 0)
```

In SQLScope, insert this information (make sure to include separator (;) after each statement) and then select Run All from the Run menu. You should also save these statements into a .sql file and write it to the orders database directory.

After these statements finish running, return to DDF Ease and click on the SalesOffices table and then select the Data tab as discussed in the next step.

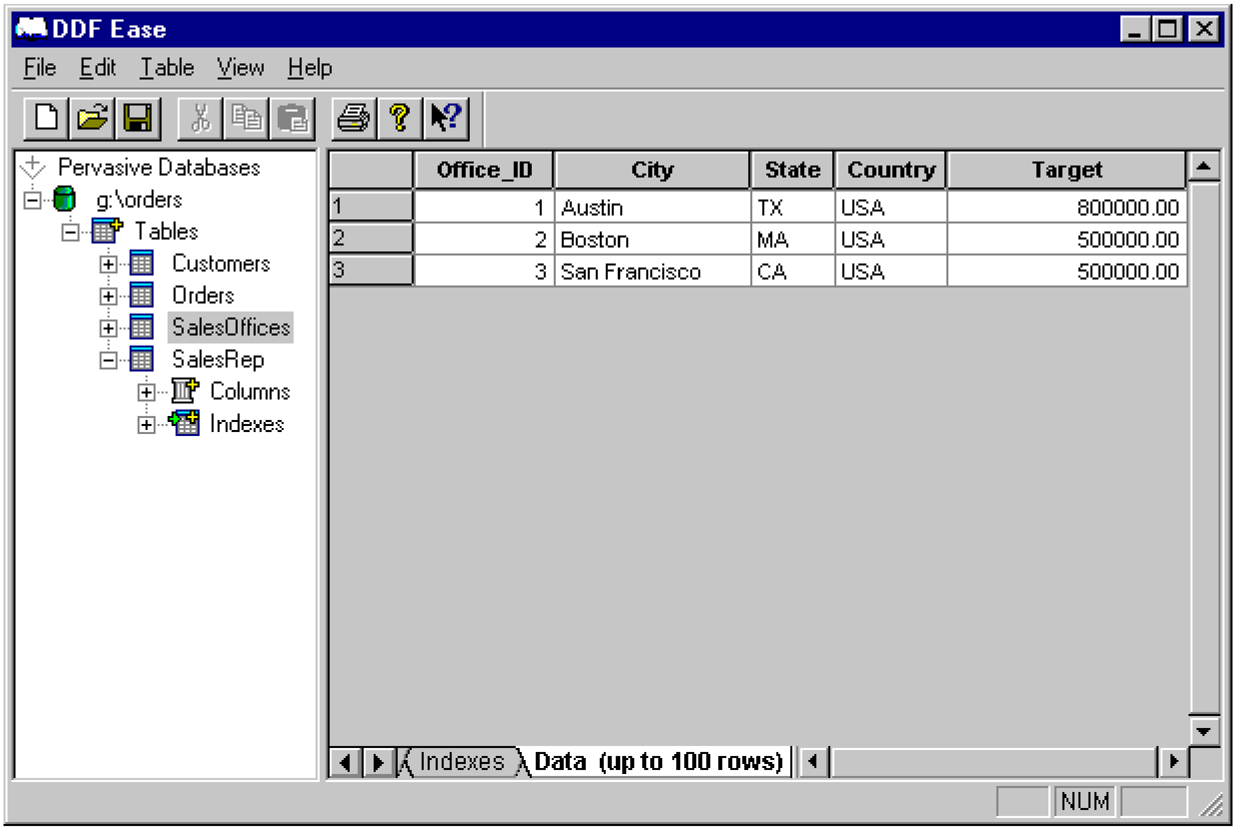
7. View Table Data.

a. DDF Ease is currently limited to displaying the first 100 rows of data for each table. To view the table data, select the table from the tree view and then select the Data tab.

For example, to view the data for the SalesOffices table that you added with SQLScope, select the SalesOffices table and then select the Data tab. If you make changes to table data and want to refresh the view, simply select another table, then reselect the table to force rereading of the data and table definitions.

Figure 4-21 displays the data we added to the SalesOffices table.

Figure 4-21
DDF Ease Main Window – Viewing Data Additions Example



	Office_ID	City	State	Country	Target
1	1	Austin	TX	USA	800000.00
2	2	Boston	MA	USA	500000.00
3	3	San Francisco	CA	USA	500000.00

Monitoring Pervasive.SQL Database Resources

This chapter includes the following sections:

- “Monitor Utility Overview”
- “Starting the Monitor Utility”
- “Setting Monitor Utility Options”
- “Monitoring MicroKernel Resources”
- “Monitoring SQL Interface Resources”

Monitor Utility Overview

The Monitor utility allows you to monitor Pervasive.SQL activities on a server. It provides information that is useful for both database administration and application programming diagnostics.

Note: The Monitor utility works only with client/server releases of Pervasive.SQL 7 or later. It is not included in the Pervasive.SQL workstation product and will not connect to it from a client requester.

The following table shows the versions of the Monitor utility and the supported platforms.

Monitor Utility	Supported Platforms
Win16	Windows NT servers Windows 3.x, Windows 9X, and Windows NT clients
Win32	Windows NT servers Windows 9X and Windows NT clients

There is also an OS/2 version of this utility that runs on an OS/2 client. However, this chapter provides instructions for the Windows version.

Starting the Monitor Utility

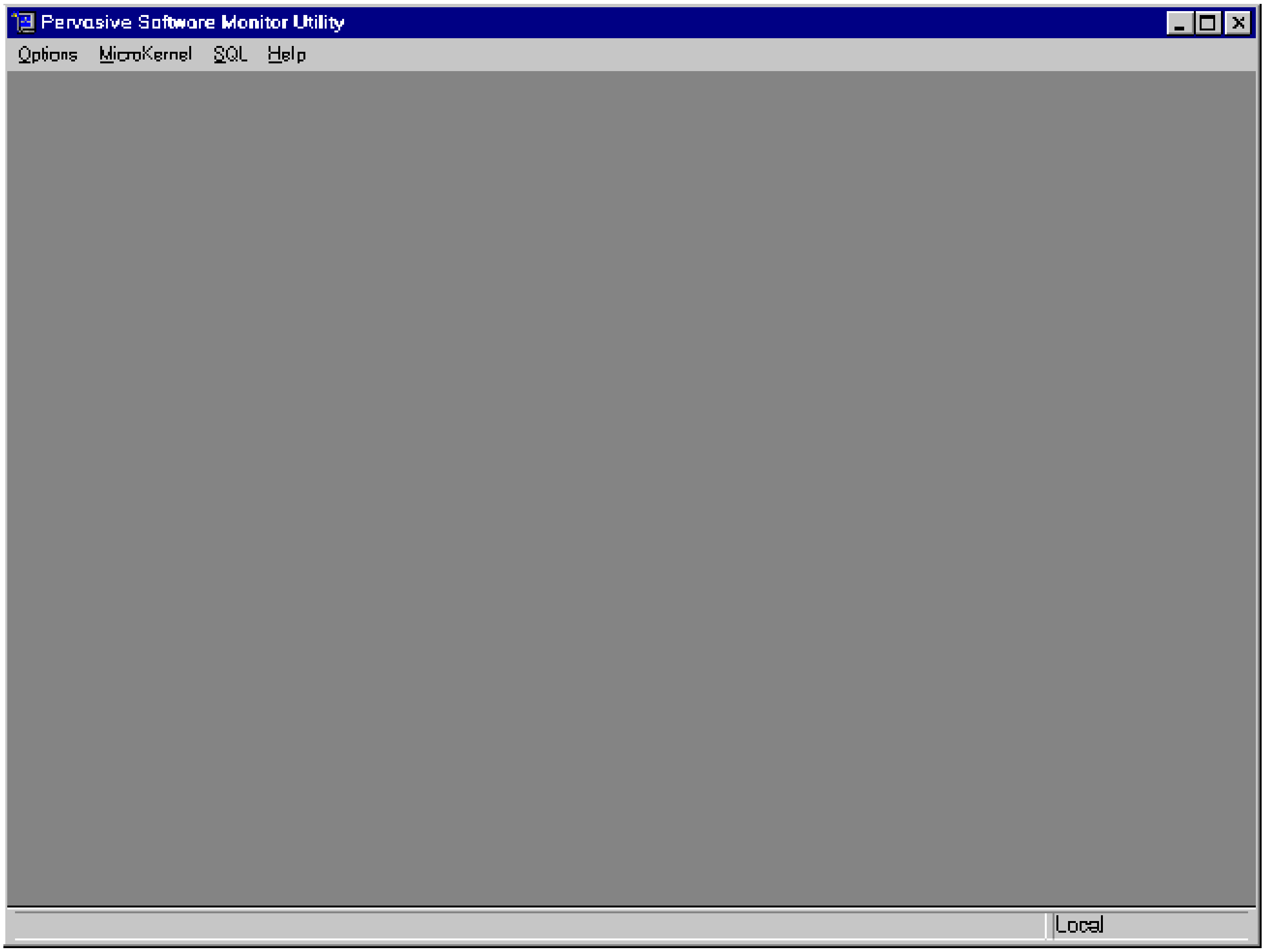
The Monitor utility provides a “snapshot” of server activity at a given point in time.

To start either the Win16 or Win32 Monitor utility from Windows 3.x or Windows 95/98/NT:

- Click Start, select Pervasive SQL 7, and then choose either Monitor (Win16) or Monitor (Win32).

  The Pervasive.SQL Monitor Utility main dialog screen is displayed.

Figure 5-1
Monitor Settings Dialog



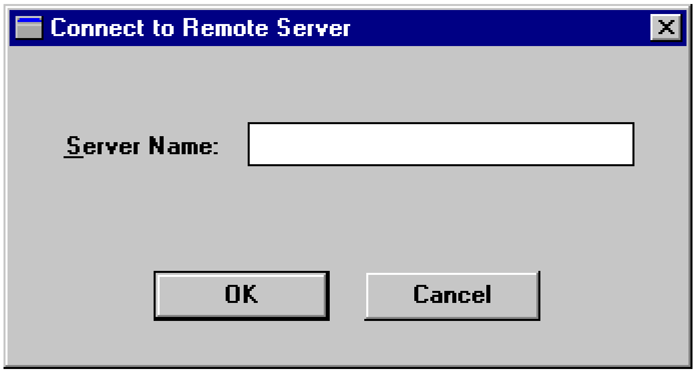
When you start the Monitor utility, it connects to the local server by default. However, you can also monitor remote server engine resources by connecting to the remote server.

Note: If dialogs are currently open in the Monitor utility window, you cannot connect to or disconnect from a remote server. Close the open dialogs before proceeding.

To connect to a remote server:

1. Choose Connect on the Options menu. The Connect to Remote Server dialog appears, as shown in Figure 5-2 on page 5-4.

Figure 5-2
Connect to Remote Server Dialog



2. Enter the server name in the Server Name box.

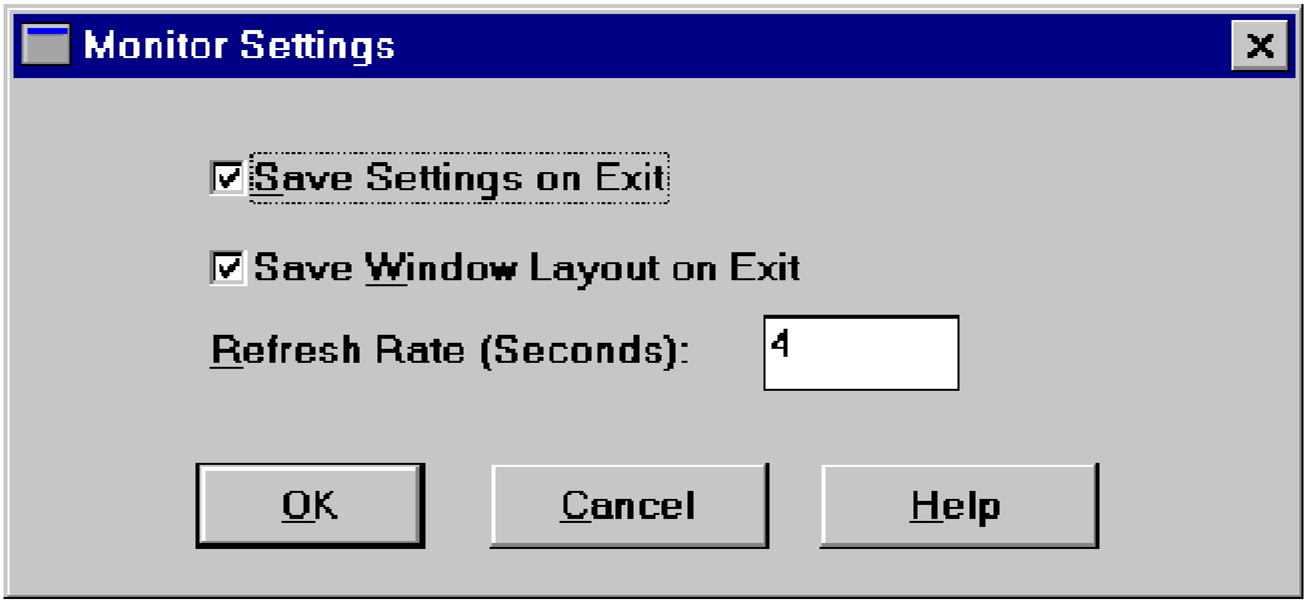
3. To disconnect from a server, choose Disconnect on the Options menu.

Setting Monitor Utility Options

To configure the Monitor utility options:

1. Choose Settings from the Options menu. The Monitor Settings dialog appears, displaying the current settings.

Figure 5-3
Monitor Settings Dialog



Monitor Settings

☑ Save Settings on Exit

☑ Save Window Layout on Exit

Refresh Rate (Seconds): 4

2. You can specify the following options:

Save Settings on Exit — Select this check box to save all configuration settings when you close the Monitor utility. The Monitor utility saves both the settings in this dialog and the automatic-refresh option in the various dialog boxes.

Save Window Layout on Exit — Select this check box to save the state (open or closed) and screen location of all open windows. When you start the Monitor utility again, these windows are automatically opened and positioned for local file server monitoring. This enables you to easily reproduce your preferred layout.

Refresh Rate (Seconds) — Specifies the frequency with which the Monitor utility's display refreshes itself. The refresh rate is measured in seconds. The default setting is 4. You can enter integer numbers only.

3. Click OK to save the settings or Cancel to close the dialog without saving changes.

Monitoring MicroKernel Resources

This section describes the following options for monitoring the MicroKernel:

- “Setting Screen Refresh Options”
- “Viewing Active Files”
- “Viewing User Information”
- “Viewing MicroKernel Resource Usage”
- “Viewing MicroKernel Communications Statistics”
-

Setting Screen Refresh Options

You can refresh the information in the Monitor utility dialogs either automatically or manually, as follows.

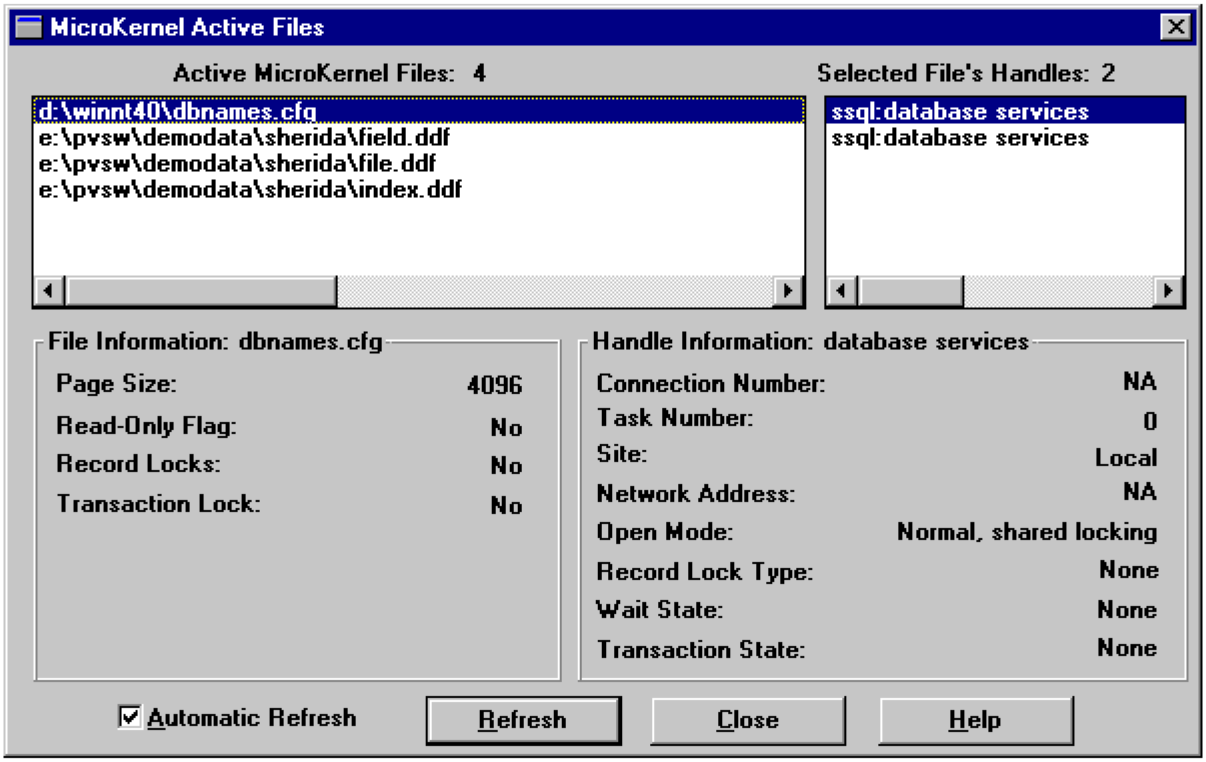
- Automatically: select the Automatic Refresh check box. The utility updates the dialogs at the Refresh Rate specified in the Monitor options (available via Settings on the Options menu).
- Manually: Click Refresh.

Viewing Active Files

To view active MicroKernel files:

- Choose Active Files from the MicroKernel menu. The MicroKernel Active Files dialog appears, as shown in Figure 5-4 on page 5-7. This dialog shows you all the active files for the MicroKernel.

Figure 5-4
MicroKernel Active Files Dialog



Active MicroKernel Files: 4

```
d:\winnt40\dbnames.cfg
e:\pvsw\demodata\sherida\field.ddf
e:\pvsw\demodata\sherida\file.ddf
e:\pvsw\demodata\sherida\index.ddf
```

Selected File's Handles: 2

```
ssql:database services
ssql:database services
```

File Information: dbnames.cfg

Page Size:	4096
Read-Only Flag:	No
Record Locks:	No
Transaction Lock:	No

Handle Information: database services

Connection Number:	NA
Task Number:	0
Site:	Local
Network Address:	NA
Open Mode:	Normal, shared locking
Record Lock Type:	None
Wait State:	None
Transaction State:	None

☑ Automatic Refresh

In the upper left of the dialog, the Monitor utility displays the Active MicroKernel Files list. This scrollable list contains the complete path of all open files in alphabetic order.

To view more information about a particular file:

- Select the desired file in the list.

In the upper right of the MicroKernel Active Files dialog, the Monitor utility displays the Selected File’s Handles list. This scrollable list contains the active handles (users) associated with the selected file. Each handle is represented by a user name (typically the login ID of the user), or by an index into the engine’s client (user) list.

To view more information about a particular user:

- Select the desired handle in the Selected File’s Handles list.

The SQL Engine handle is specified as ssql:scalable sql. The Database Services client handle is specified as ssql:database services. SQL Interface logins and Database Services logins are specified as ssql:username or ssql:sessionNumber. Some handles have an agent identifier, a two letter code that specifies the application that initiated the session.

Table 5-1 lists the agent IDs used by Pervasive.SQL components.

Table 5-1
Agent IDs

Agent ID	Application or Component

BT	Maintenance utility for the Btrieve Interface (BUTIL)
DE	Database Services client
DC	Database Services login
DR	DOS client Requester
ML	MicroKernel logging and roll forward
NR	Windows 95 and Windows NT client Requester
NX	Maintenance utility for the SQL Interface (SQLUTIL)
OR	OS/2 client Requester
PU	Pervasive.SQL utilities
RU	RI Utility (RIUTIL)
SC	SQL Interface login
SE	SQL Engine
WR	Windows client Requester

The File Information box displays detailed information about the selected file. The Handle Information box displays detailed information about the selected handle.

File Information

The File Information box displays the following information about each file:

Page Size Indicates the size in bytes of each page in the file.

Read-Only Flag Indicates whether the file is flagged as read-only by the operating system.

Record Locks Indicates whether any of the active handles for the selected file have record locks. Any application can read a locked record, but only the application that placed the lock can modify or delete the record. A record lock exists only as long as the application that opened the file is updating a record. Yes indicates that one or more record locks are applied to the file; No indicates that no records are locked.

Transaction Lock Indicates whether any of the active handles for the selected file have a transaction lock. A transactional file lock exists only as long as the application that opened the file is processing a transaction.

Handle Information

The Handle Information box displays the following information about each file:

Connection Number Displays the network connection number of the client. If the client does not have a network connection, this field displays NA (for not applicable).

Task Number Displays the process-supplied task number for processes originating at the server, a Windows client, or an OS/2 client. If the process originates at a DOS client, this field contains the communications protocol socket number.

Site

Specifies the location of the user process (local or remote).

Network Address

Identifies the location of the calling process on the network. If the calling process is SPX, then network node/network address is preceded by S: such as S: 65666768 00000000001. If the calling process is TCP/IP, the dotted-decimal notation of the IP number is preceded by T: such as T: 180.150.1.24.

Open Mode

Indicates the method the application uses to open the specified handle of the file. Valid open modes are:Normal—The application that opened the file has normal shared, read/write access to it.

Accelerated—The application that opened the file has shared read/write access.

Read-only—The application that opened the file has read-only access; it cannot modify the file.

Exclusive—The application that opened the file has exclusive access. Other applications cannot open the file until the calling application closes it.

Verify—The application that opened the file also ensures that the operating system stores all write operations in a file.

The Monitor utility also specifies all open modes as non-transactional or shared locking when applicable.

Record Lock Type

Displays the type of record lock(s) currently held by the handle. The possible record lock types are Single, Multiple, and None.

Single-record locks enable a user to lock only one record at a time. Multiple-record locks enable a user to lock more than one record at a time.

Wait State

Indicates whether the user is waiting due to some type of lock on this handle: Waits for Record Lock, Waits for File Lock, or None.

Transaction State

Displays the state of the transaction lock currently held by the handle. The possible transaction types are Exclusive, Concurrent, or None.

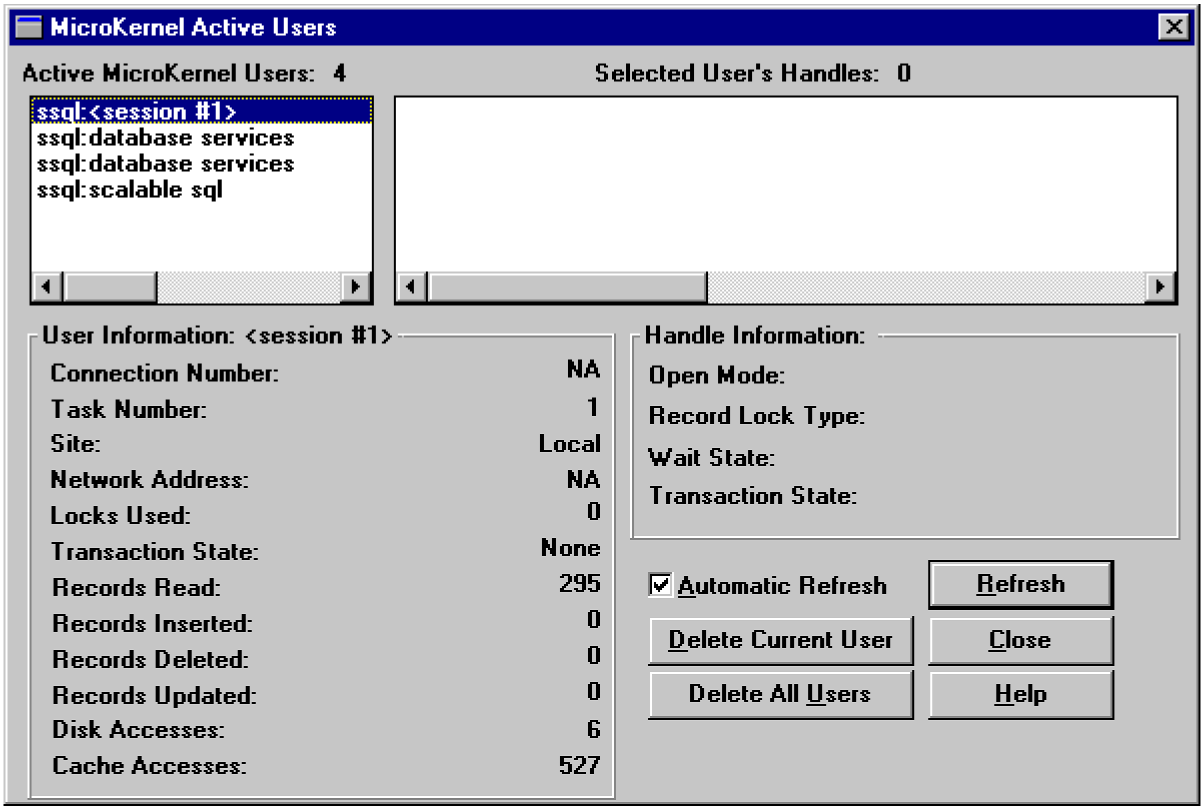
Viewing User Information

You can view a list of current users and files, as well as file handles for each user.

To view MicroKernel user information:

- Choose Active Users on the MicroKernel menu. The MicroKernel Active Users dialog appears, as shown in Figure 5-5 on page 5-11.

Figure 5-5
MicroKernel Active Users Dialog


MicroKernel Active Users

Active MicroKernel Users: 4

Selected User's Handles: 0

ssql:<session #1>
ssql:database services
ssql:database services
ssql:scalable sql

User Information: <session #1>

Connection Number:	NA
Task Number:	1
Site:	Local
Network Address:	NA
Locks Used:	0
Transaction State:	None
Records Read:	295
Records Inserted:	0
Records Deleted:	0
Records Updated:	0
Disk Accesses:	6
Cache Accesses:	527

Handle Information:

Open Mode:

Record Lock Type:

Wait State:

Transaction State:

☑ Automatic Refresh

In the upper left of the dialog, the Monitor utility displays the Active MicroKernel Users list. This scrollable list contains the names of active users in alphabetic order. Each user is represented by a user name (typically the login ID of the user) or by an index into the engine’s client (user) list.

To receive more information about a particular user:

- Highlight the desired user in the list.

Each client is represented by either a user name (typically the login ID of the user) or an index into the engines client (user) list. Table 5-1 lists the agent IDs used by Pervasive.SQL components.

In the upper right of the MicroKernel Active Users dialog, the Monitor utility displays the Selected User’s Handles list. This scrollable list contains the active handles (files) associated with the selected user. The MicroKernel creates a handle each time a user opens a file; therefore, a single user can have several handles for the same file.

To view more information about a particular file:

- Highlight the desired handle in the list.

The User Information box displays the following detailed information for the selected user file handle (for a description of the Connection Number, Task Number, Site, and Network Address fields and the Handle Information box, refer to “Handle Information”):

Locks Used
Indicates the number of locks the user is currently using.

Transaction State
Displays the type of transaction lock the user currently holds. The possible transaction types are Exclusive, Concurrent, or None.

Records Read
Displays the number of records read since the user first opened a file.

Records Inserted
Displays the number of records the user has inserted.

Records Deleted
Displays the number of records the user has deleted.

Records Updated
Displays the number of records the user has updated.

Disk Accesses
Indicates the number of times the user required a disk access. You will not see any information for disk accesses for files that have just been opened.

Cache Accesses
Displays the number of times the user required a cache access.

Deleting Current Users

To delete a user:

- Highlight the user name and click Delete Current User button. Deleting the current user removes the user from the list of active users of the MicroKernel and terminates the user’s connection to the Communications Server.

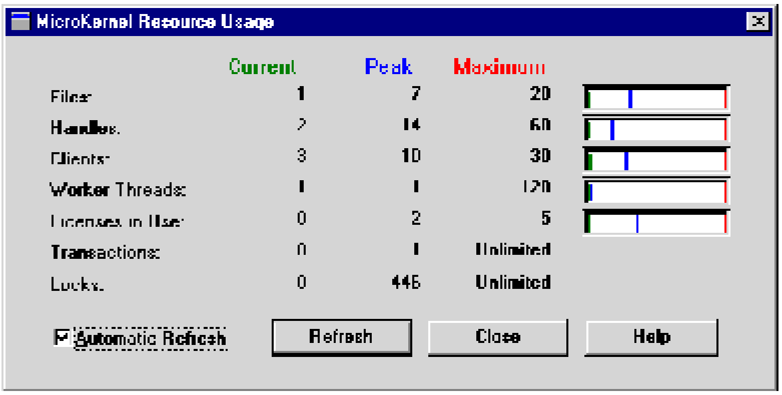
- You can also click Delete All Users which deletes all of the current MicroKernel users.

Viewing MicroKernel Resource Usage

To view MicroKernel resource usage:

1. Choose Resource Usage from the MicroKernel menu. The MicroKernel Resource Usage dialog appears, as shown in Figure 5-6 on page 5-13.

Figure 5-6
MicroKernel Resource Usage Dialog



	Current	Peak	Maximum
Files:	1	7	20
Handles:	2	14	60
Clients:	3	10	30
Worker Threads:	1	1	120
Licenses in Use:	0	2	5
Transactions:	0	1	Unlimited
Locks:	0	446	Unlimited

☑ Automatic Refresh

This dialog allows you to view the total resources in use by the MicroKernel since it was loaded.The MicroKernel Resource Usage dialog shows the following statistics for each resource:

- Current – Shows the present value for the field.
- Peak – Shows the highest value for the field since the MicroKernel was started.
- Maximum – Shows the highest value allowed for the field.

Files
Indicates the number of active files. You set the maximum for this field with the Setup utility (see "Open Files" on page 3-39).

Handles
Indicates the number of active handles. The MicroKernel creates a handle each time a user opens a file; therefore, a single user can have several handles for the same file. You set the maximum for this field with the Setup utility (see "Handles" on page 3-40).

Clients
Indicates the number of clients accessing the MicroKernel. A workstation can have multiple clients accessing the engine simultaneously. You set the maximum for this field with the Setup utility (see "Active Clients" on page 3-50).

I/O Threads
Indicates the number of concurrent MicroKernel processes. You configure this setting with the Setup utility (see "I/O Threads" on page 3-50).

Licenses in Use
Indicates the number of users of the Btrieve interfaces as defined by your licensing agreement. In this case, the maximum shows the number of users your licensing agreement allows.

Transactions
Indicates the number of transactions. The maximum for this field is unlimited.

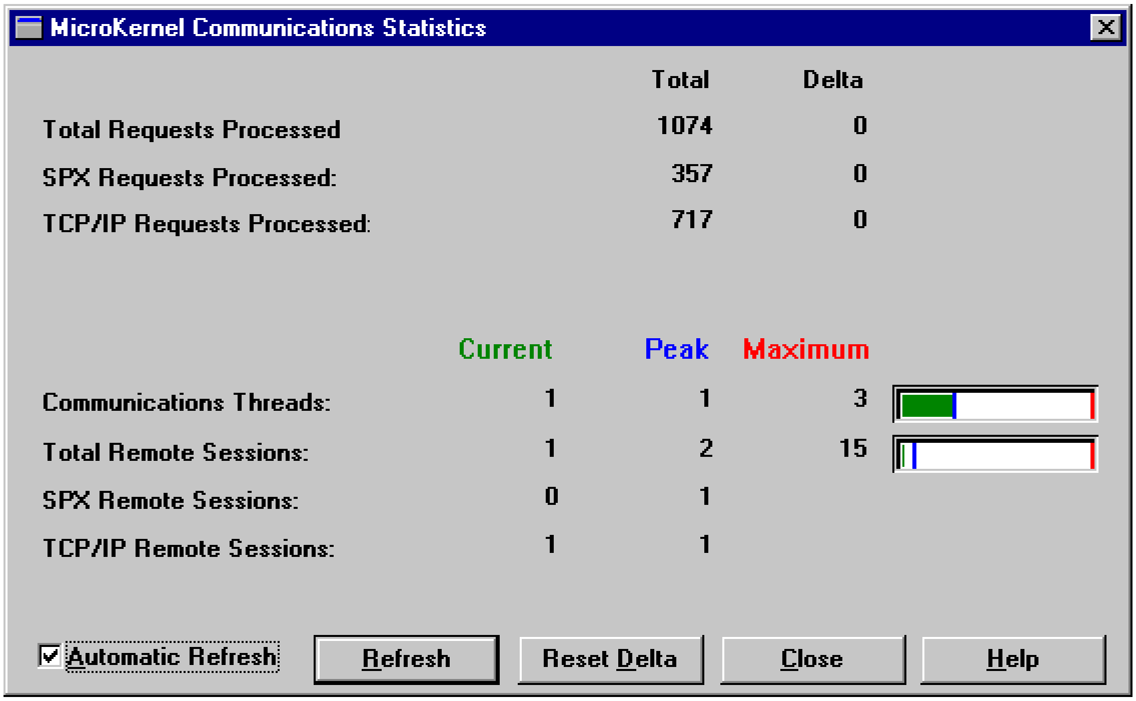
Locks
Indicates the number of record locks. The maximum for this field is unlimited.

Viewing MicroKernel Communications Statistics

To view MicroKernel communications statistics:

1. Choose Communications from the MicroKernel menu. The MicroKernel Communications Statistics dialog appears, as shown in Figure 5-7 on page 5-15. This dialog shows you the network requests, worker threads, and sessions in use by the Communications Server since it was loaded.

Figure 5-7
MicroKernel Communications Statistics Dialog



	Total	Delta
Total Requests Processed	1074	0
SPX Requests Processed:	357	0
TCP/IP Requests Processed:	717	0

	Current	Peak	Maximum
Communications Threads:	1	1	3
Total Remote Sessions:	1	2	15
SPX Remote Sessions:	0	1	
TCP/IP Remote Sessions:	1	1	

☑ Automatic Refresh

The MicroKernel Communications Statistics dialog shows the following statistics for several of the communications resources:

- Current – Shows the present value for each field.
- Peak – Shows the highest value for the field since the Communications Manager was started.
- Maximum – Shows the highest value allowed for the field.

You can monitor the activity of the following communications resources in the MicroKernel Communications Statistics dialog:

Total Requests Processed
Indicates the number of requests the Communications Manager handles from workstations or remote, server-based applications.

SPX Requests Processed
Indicates the number of SPX requests the Communications Manager handles from clients or remote, server-based applications.

TCP/IP Requests Processed
Indicates the number of TCP/IP requests the Communications Manager handles from clients or remote, server-based applications.

Total – Indicates the number of requests processed since the Communications server was loaded.

Delta – Indicates the number of requests since you first invoked the Communications Statistics dialog. To reset this number to zero, click Reset Delta.

Worker Threads

Indicates the number of remote requests that the MicroKernel is currently processing. Local requests are not included in this statistic. For the total number of remote and local threads being processed, see the Resource Usage dialog. You set the maximum for this field with the Setup utility (the Number of Communications Threads option).

Worker threads are also used to process Monitor utility requests, so you may not see the number of current worker threads drop below one. This is normal.

Remote Sessions

Indicates the number of remote clients connected to the Communications Manager. You set the maximum for this field with the Setup utility (the Number of Sessions option).

SPX Remote Sessions

Indicates the number of remote clients connected via SPX to the Communications Manager.

TCP/IP Remote Sessions

Indicates the number of remote clients connected via TCP/IP to the Communications Manager.

Monitoring SQL Interface Resources

This section describes the following options for monitoring the SQL Engine:

- “Setting Screen Refresh Options”
- “Viewing Active SQL Sessions” on page 5-17
- “Viewing SQL Resource Usage” on page 5-21
- “Viewing SQL Communications Statistics” on page 5-23

Setting Screen Refresh Options

You can refresh the information the Monitor utility dialogs either automatically or manually, as follows.

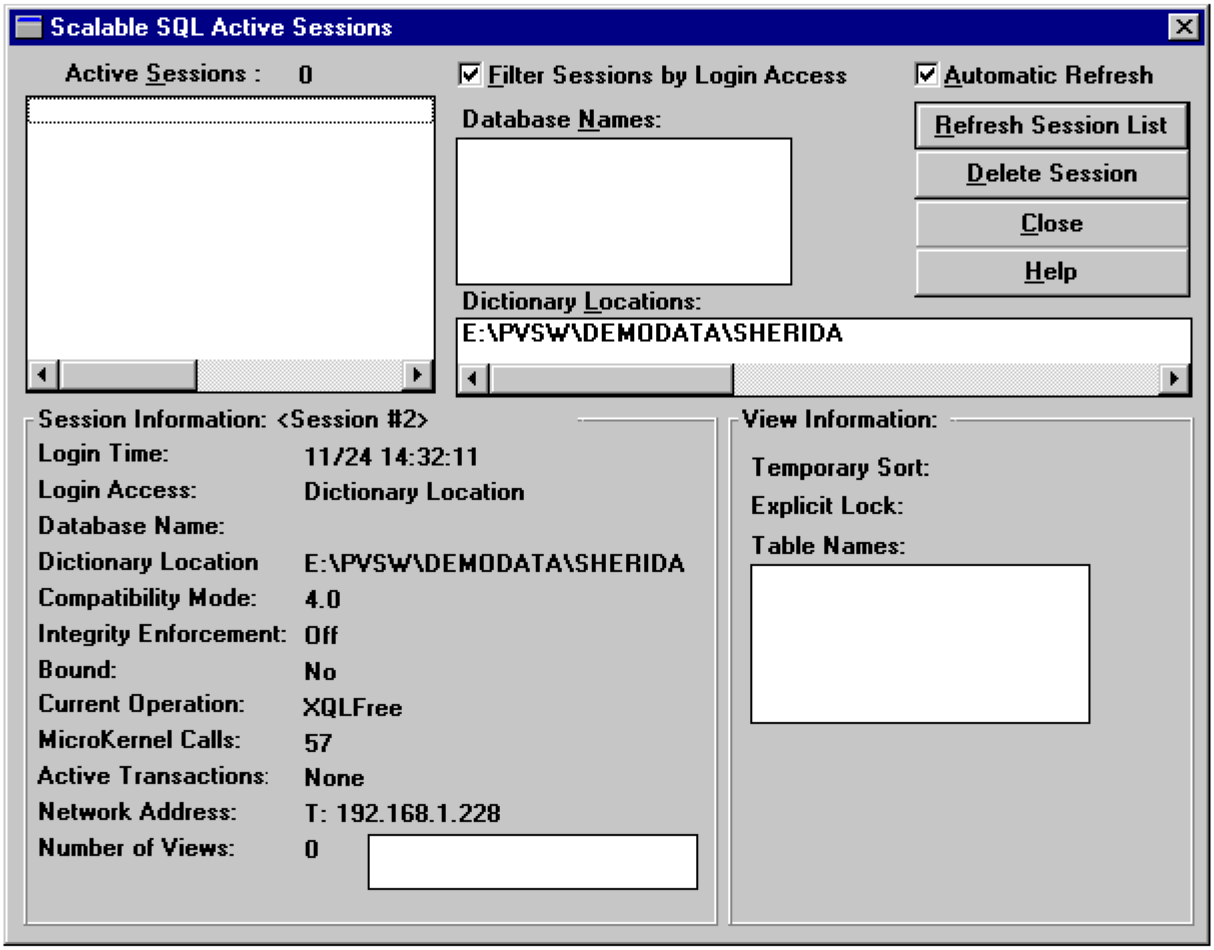
- Automatically: select the Automatic Refresh check box. The utility updates the dialogs at the Refresh Rate specified in the Monitor options (available via Settings on the Options menu).
- Manually: Click Refresh.

Viewing Active SQL Sessions

To view active SQL Interface sessions:

- Choosing Active Sessions from the SQL menu. The Scalable SQL Active Sessions dialog appears, as shown in Figure 5-8 on page 5-18.

Figure 5-8
Scalable SQL Active Sessions Dialog



Active Sessions : 0

☑ Filter Sessions by Login Access

☑ Automatic Refresh

Database Names:

Dictionary Locations: E:\PVSW\DEMODATA\SHERIDA

Session Information: <Session #2>

Login Time:	11/24 14:32:11
Login Access:	Dictionary Location
Database Name:	
Dictionary Location	E:\PVSW\DEMODATA\SHERIDA
Compatibility Mode:	4.0
Integrity Enforcement:	Off
Bound:	No
Current Operation:	XQLFree
MicroKernel Calls:	57
Active Transactions:	None
Network Address:	T: 192.168.1.228
Number of Views:	0

View Information:

Temporary Sort:

Explicit Lock:

Table Names:

This dialog provides information for any of the active SQL Interface sessions, as well as information about the views a selected session is currently accessing. You can filter the list of sessions either by the database name or by the dictionary location.

Filtering Active Sessions

To filter the active sessions:

1. Select the Filter Sessions by Login Access check box.

   Select all sessions either from one database name or from one dictionary.

2. Click the database name in the Database Names list or the dictionary location in the Dictionary Locations list, and the corresponding active sessions appear in the Active Sessions list. (Dictionary locations list the full path and appear in alphabetic order.)

3. If you clear the Filter Sessions by Login Access check box, the dialog displays all sessions and disables the database name and dictionary location controls.

Session Information

The top of the Active Sessions box displays the number of active SQL Interface sessions for the currently selected user name in the Active Sessions list. If you applied a filter, the number of active sessions displayed pertains only to the specified database name or dictionary. (A session is formed when an application or task logs in to a database.)

The Session Information box provides the following information about the currently selected session in the Active Sessions list.

Login Time
Specifies when the session was established via a login operation. The time appears in the following format: month/day followed by hour:minute:second.

Login Access
Indicates the access path for this session’s login. This field displays either Database Name or Dictionary Location.

Database Name
Displays the database name, if the user logged into this session either using a database name or using a dictionary path to a named database.

Dictionary Location
Displays the dictionary location, if the user logged into this session using a dictionary path to either an unnamed database or a database that has multiple names. This field is empty if the user logged in either using a database name or a dictionary path to a named database.

Compatibility Mode
Displays the compatibility mode being used in the current session. Possible values are 3.01 (for SQL Interface v3.01 compatibility) or 4.0. For more information about compatibility modes, refer to the What’s New.

Integrity Enforcement
Indicates whether Pervasive.SQL is enforcing integrity constraints (including security, referential integrity, and triggers) on the database. To use the integrity enforcement option, refer to ”Maintain Named Databases” on page 3-15.

Bound
Indicates whether the database accessed by the current session is a bound database. For more information about bound databases, refer to the Pervasive.SQL Programmer’s Guide.

Current Operation
Indicates the SQL API function that this session called most recently.

MicroKernel Calls
Indicates the number of function calls that this session has made to the MicroKernel.

Active Transactions
Indicates whether this session is currently within a transaction.

Network Address
Identifies the location of the calling process on the network. If the calling process is SPX, then network node/network address is preceded by S: such as S: 65666768 00000000001. If the calling process is TCP/IP, the dotted-decimal notation of the IP number is preceded by T: such as T: 180.150.1.24. For local access, this field displays Local.

Number of Views
Indicates the number of active views, which are shown in the list box.

View Information

The View Information box displays current information about all the views the selected session is accessing. It provides the following information about the currently selected view in the Number of Views list.

Temporary Sort	Indicates whether a temporary sort order is active for the view.
Explicit Lock	Indicates whether explicit locks have been issued for the view.
Table Names	Lists the number of tables and the name of each table in the view.

Refreshing Session List

To periodically refresh the session listing:

- Click Refresh Session List whether or not the Automatic Refresh is enabled.

Deleting SQL Interface Sessions

To delete a SQL Interface session:

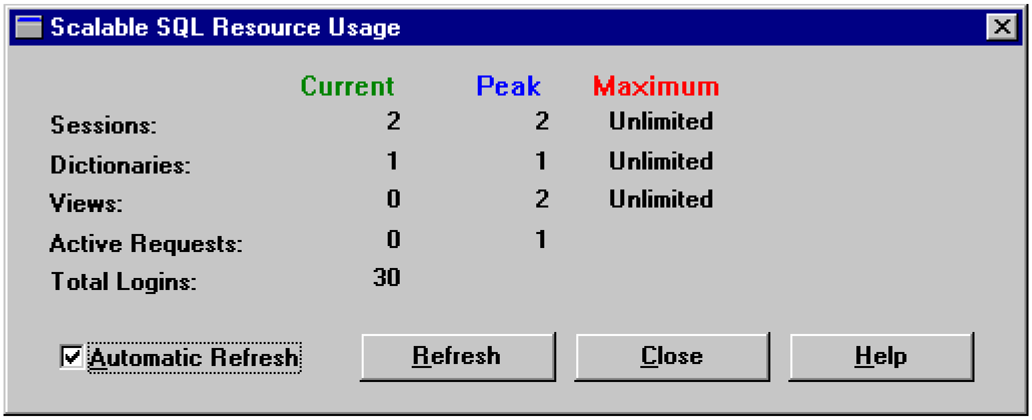
- Click Delete Session; the SQL Engine kills the session you delete.

Viewing SQL Resource Usage

To view SQL Interface resource usage:

1. Choose Resource Usage from the SQL menu. The Scalable SQL Resource Usage dialog appears, as shown in Figure 5-9 on page 5-21.

Figure 5-9
Scalable SQL Resource Usage Dialog



	Current	Peak	Maximum
Sessions:	2	2	Unlimited
Dictionaries:	1	1	Unlimited
Views:	0	2	Unlimited
Active Requests:	0	1	
Total Logins:	30		

☑ Automatic Refresh

The Scalable SQL Resource Usage dialog shows the following statistics for each resource:

- Current – Shows the present value for the field.
- Peak – Shows the highest value for the field since the SQL Engine was started.
- Maximum – Shows the highest value allowed for the field.

A horizontal bar chart for some of the following fields shows the current and peak values in relation to the maximum value; the Monitor utility does not display a bar chart if the maximum value for a field is unlimited.

Sessions
Indicates the total number of active SQL Interface sessions. Each session corresponds to a login operation.

Dictionaries
Indicates the number of databases with at least one active session. The maximum number of dictionaries is unlimited.

Views
Indicates the total number of active views. The maximum number of views is unlimited.

Active Requests
Indicates the number of concurrent requests being processed.

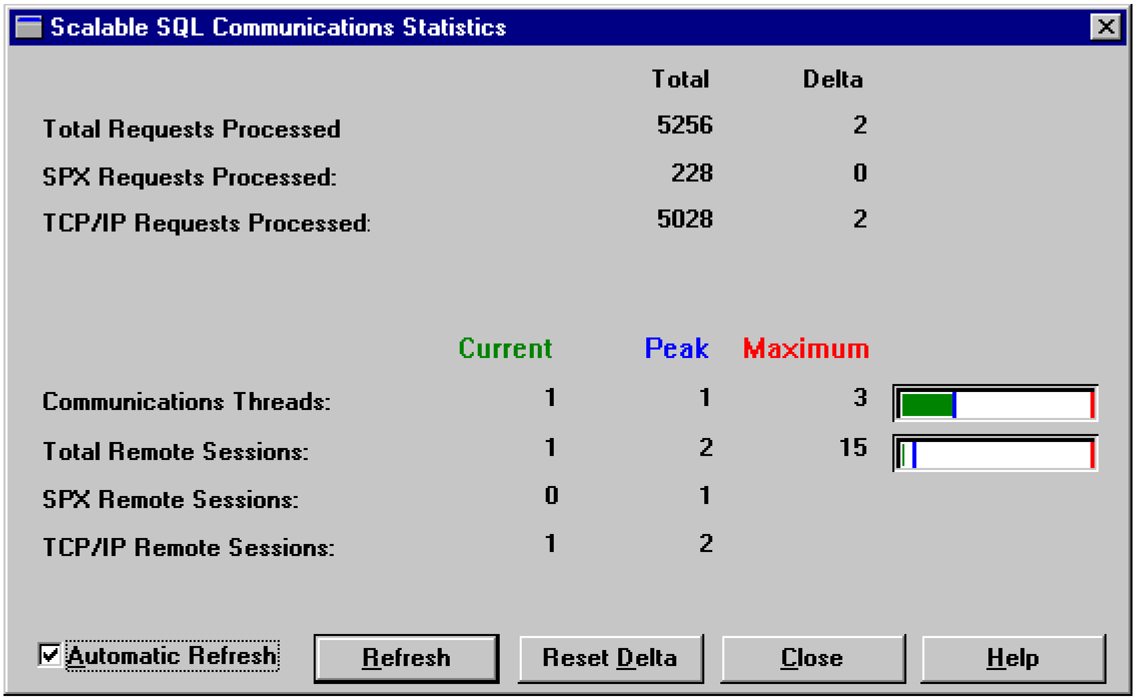
Total Logins
Indicates the total number of sessions since the SQL Engine was loaded. Each login operation corresponds to a session.

Viewing SQL Communications Statistics

To view SQL Interface communications statistics:

- Choose Communications on the SQL menu. The Scalable SQL Communications Statistics dialog appears.

Figure 5-10
Scalable SQL Communications Statistics Dialog



Scalable SQL Communications Statistics

	Total	Delta
Total Requests Processed	5256	2
SPX Requests Processed:	228	0
TCP/IP Requests Processed:	5028	2

	Current	Peak	Maximum
Communications Threads:	1	1	3
Total Remote Sessions:	1	2	15
SPX Remote Sessions:	0	1	
TCP/IP Remote Sessions:	1	2	

☑ Automatic Refresh

This dialog displays the following information:

Requests Processed
Indicates the number of SQL requests that the communications component handled since it was loaded. Delta indicates the count since you first invoked the dialog or the delta was reset.

SPX Requests Processed
Indicates the number of SPX requests the Communications Manager handles from clients or remote, server-based applications.

TCP/IP Requests Processed
Indicates the number of TCP/IP requests the Communications Manager handles from clients or remote, server-based applications.

Total – Indicates the number of requests processed since the Communications server was loaded.

Delta – Indicates the number of requests since you first invoked the Communications Statistics dialog. To reset this number to zero, click Reset Delta.

Worker Threads
Indicates the number of remote requests that the SQL Engine is currently processing.

Also, local requests being processed may not be included in this statistic. See the Scalable SQL Resource Usage Dialog for the total (remote and local) number of threads being processed.

Worker threads are also used to process Monitor utility requests, so you may not see

	the number of current worker threads drop below one. This is normal.
Remote Sessions	Indicates the number of remote workstations connected to the communications component.
SPX Remote Sessions	Indicates the number of remote clients connected via SPX to the Communications Manager.
TCP/IP Remote Sessions	Indicates the number of remote clients connected via TCP/IP to the Communications Manager.

Resetting the Delta Count

To reset the delta count to zero:

- Click Reset Delta.

Testing Btrieve Operations Using the Function Executor

This chapter discusses the following topics:

- "Function Executor Overiew"
- "Starting the Function Executor Utility"
- "Overview of the Function Executor Main Window"
- "Editing the Key or Data Buffer (OS/2 Clients only)"
- "Performing Operations"

Function Executor Overview

The 16-bit Function Executor runs on Win16, Win32 and OS/2 platforms for Pervasive.SQL server and workstation products. With this interactive utility, you can learn how Btrieve operations work. By allowing you to execute Btrieve operations one at a time, the Function Executor enables application developers to simulate the operations of a Btrieve application, which can help in testing and debugging your program.

The Function Executor is primarily a tool for application developers; this chapter assumes a basic knowledge of Btrieve operations. For more information about Btrieve operations, refer to the Pervasive.SQL Programmer's Reference.

Starting the Function Executor Utility

To start the Function Executor utility on Win16 platforms:

- Double-click the Function Executor icon in the PVSW/Bin program group.

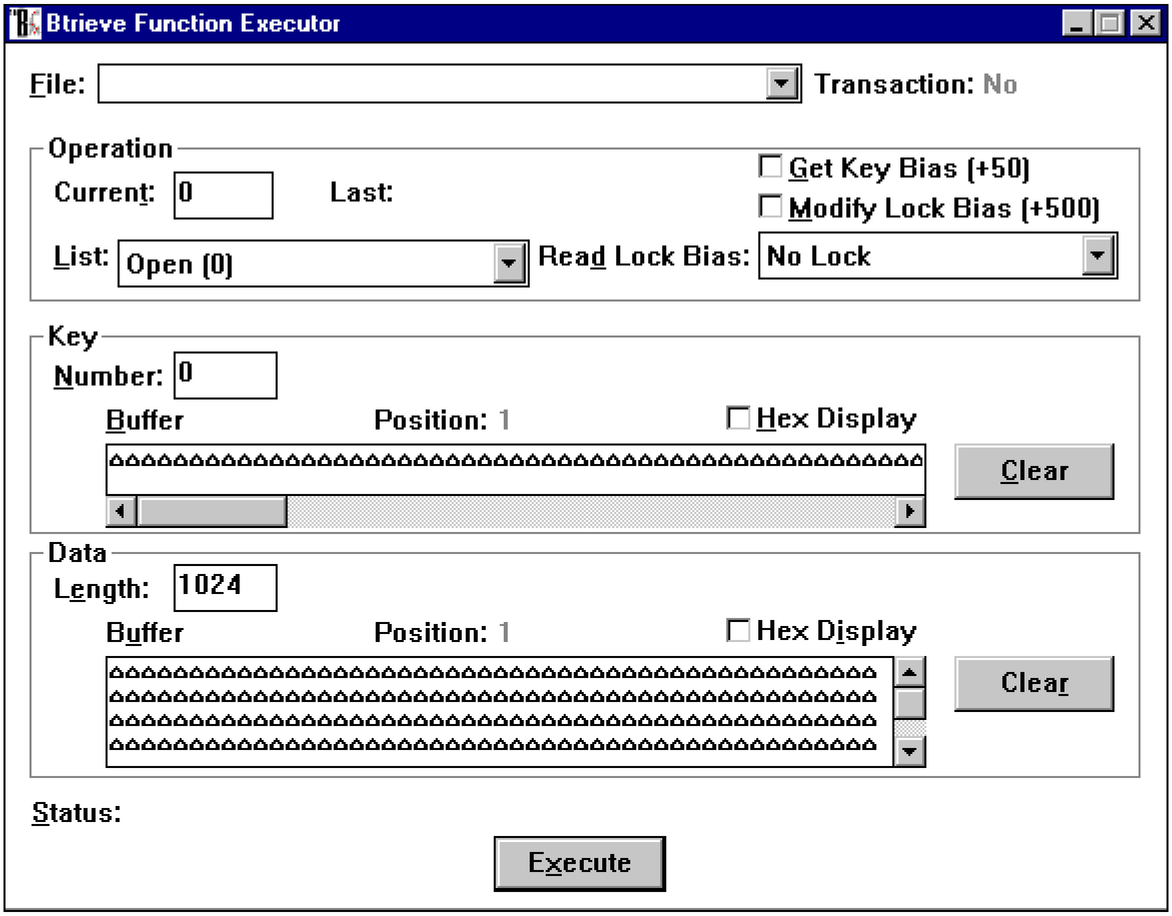
To start the Function Executor utility on Win32 platforms:

1. From the Start menu, select Programs and then Pervasive SQL 7.
2. Select Function Executor (Win16). The main window (Figure 6-1) appears.

To start the Function Executor utility on an OS/2 client:

1. From the Start menu, select Pervasive SQL 7.
2. Select Function Executor (Win16). The main window (Figure 6-2) appears.

Figure 6-1
Function Executor Main Window



Btrieve Function Executor

File:

Transaction: No

Operation

Current: 0

Last:

□ Get Key Bias (+50)

□ Modify Lock Bias (+500)

List: Open (0)

Read Lock Bias: No Lock

Key

Number: 0

Buffer

Position: 1

□ Hex Display

Data

Length: 1024

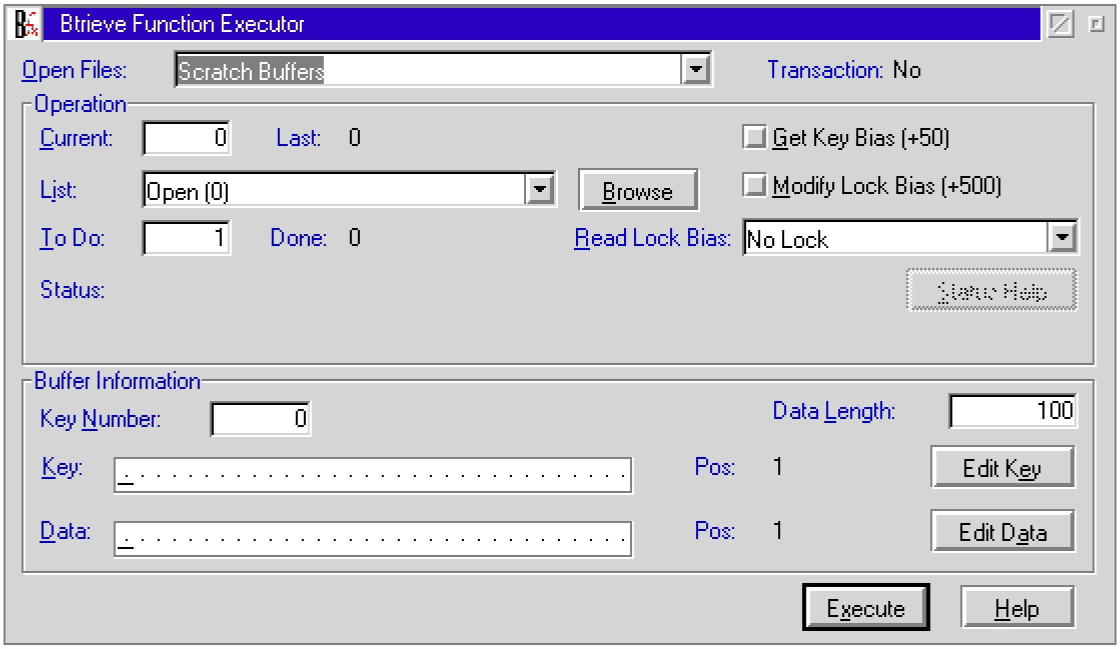
Buffer

Position: 1

□ Hex Display

Status:

Figure 6-2
Function Executor Main Window (OS/2 version)



Btrieve Function Executor

Open Files: Scratch Buffers

Transaction: No

Operation

Current: 0

Last: 0

□ Get Key Bias (+50)

□ Modify Lock Bias (+500)

List: Open (0)

To Do: 1

Done: 0

Read Lock Bias: No Lock

Status:

Buffer Information

Key Number: 0

Data Length: 100

Key:

Pos: 1

Data:

Pos: 1

Overview of the Function Executor Main Window

Table 6-1 lists the controls in the Win16 main window. Some of the controls correspond to Btrieve Interface function parameters.

Table 6-1
Function Executor Controls for Win16

Control	Description
File	Lists the full path of all open data files and displays the current open file. You can move among open files, but you cannot open a file using this box. To open a file, refer to “Opening a File”.This control corresponds with the Position Block parameter. Because each file name represents a position block, a file name can appear more than once in the list if the file has been opened more than once.
Transaction	Indicates whether the current operation occurs inside a transaction and the type of transaction, as follows: Exclusive – Exclusive transaction. Concurrent – Concurrent transaction. Conc+ModLk – Concurrent transaction with Modify Lock (+500) bias.
Operation	
Current	Specifies the current operation code plus its bias (if any). The default is 0. If you are familiar with Btrieve operation codes, you can enter the desired code. Otherwise, use the List box to specify an operation. This control corresponds with the Operation Code parameter.
Last	Displays the code of the last operation that was executed on the current file.
List	Lists all Btrieve operations and their codes. The default is Open (0). You can move quickly through the list by entering the first letter of the operation you want to perform.
Get Key Bias (+50)	Instructs the MicroKernel to return only a key value, not a data record, on the current Get operation.
Modify Lock Bias (+500)	Instructs the MicroKernel to set a no-wait lock bias on an insert, update, or delete operation executed within a concurrent transaction.
Read Lock Bias	Adds one of five biases to the current operation, as follows. For files in exclusive transactions, the MicroKernel ignores any lock bias values you specify explicitly. No Lock – Performs no locking. (Default) Single Wait (+100) – Attempts to lock a single record; if the record is already locked, it waits until the record is free. Single No Wait (+200) – Attempts to lock a single record and returns control if the record is already locked. Multiple Wait (+300) – Attempts to lock multiple records in the same file; if the records are already locked, it waits until the records are free. Multiple No Wait (+400) – Attempts to lock multiple records in the same file and returns control if the records are already locked.
Key	
Number	For most Get operations, specifies a key number, or index path, to follow for the current operation. For other operations, specifies such information as file open mode, encryption, or logical disk drive. This control corresponds with the Key Number parameter.
Buffer	Specify the path for the data file for which you want to perform a Btrieve operation.

Position	Indicates the position of the cursor within the Key buffer.
Hex Display	Click this check box to view the data in Hex format.
Clear	Click this button to clear the buffer field so that you can enter another data file.
Data	
Length	Specifies the length (in bytes) of the Data Buffer. The default is 100. For every operation that requires a data buffer, you must specify a buffer length. On many operations, the MicroKernel returns a value to the Data Length. Generally, you should always specify a Data Length before you execute an operation. This control corresponds with the Data Buffer Length parameter.
Buffer	Specifies a data value. For read and write operations, the Data Buffer contains records. For other operations, the Data Buffer contains file specifications, filtering conditions, and other information the MicroKernel needs for processing the operation. This control corresponds with the Data Buffer parameter.
Position	Indicates the position of the cursor within the Key or Data Buffer.
Hex Display	Click this check box to view the data in Hex format.
Status	Displays a numeric status code returned by the MicroKernel and a brief message explaining the result of a Btrieve operation. For detailed information about these status codes and messages, refer to the Status Codes and Messages manual.
Execute	Performs the currently specified operation.

Note: The Win16 version of the utility performs wait lock simulation. Win16 applications cannot go into a wait loop. If you ask for a record lock using a Wait Bias and the record is locked by someone else, the engine returns Status Code 84 or 85 immediately to the application.

Win16 Function Executor utility retries the operation until it gets the record or you click Abort which is displayed in the lower right corner of the Main window.

Table 6-2 on page 6-8 lists the controls in the OS/2 main windows.

Table 6-2

Function Executor Controls for OS/2

Control	Description
Open File	Lists the full path of all open data files and displays the current open file. You can move among open files, but you cannot open a file using this box. To open a file, refer to “Opening a File”. This control corresponds with the Position Block parameter. Because each file name represents a position block, a file name can appear more than once in the list if the file has been opened more than once. In the Open Files list, the Scratch Buffers entry corresponds to a set of buffers that are always available, but never updated by Btrieve operations. Use this set of buffers to open additional files without affecting the key buffer of an already open file.
Transaction	Indicates whether the current operation occurs inside a transaction and the type of transaction, as follows:

Exclusive – Exclusive transaction.
Concurrent – Concurrent transaction.
Conc+ModLk – Concurrent transaction with Modify Lock (+500) bias.

Operation

Current
Specifies the current operation code plus its bias (if any). The default is 0. If you are familiar with Btrieve operation codes, you can enter the desired code. Otherwise, use the List box to specify an operation. This control corresponds with the Operation Code parameter.

Last
Displays the code of the last operation that was executed on the current file.

List
Lists all Btrieve operations and their codes. The default is Open (0). You can move quickly through the list by entering the first letter of the operation you want to perform.

Get Key Bias (+50)
Instructs the MicroKernel to return only a key value, not a data record, on the current Get operation.

Modify Lock Bias (+500)
Instructs the MicroKernel to set a no-wait lock bias on an insert, update, or delete operation executed within a concurrent transaction.

Read Lock Bias
Adds one of five biases to the current operation, as follows. For files in exclusive transactions, the MicroKernel ignores any lock bias values you specify explicitly.

No Lock – Performs no locking. (Default)
Single Wait (+100) – Attempts to lock a single record; if the record is already locked, it waits until the record is free.
Single No Wait (+200) – Attempts to lock a single record and returns control if the record is already locked.
Multiple Wait (+300) – Attempts to lock multiple records in the same file; if the records are already locked, it waits until the records are free.
Multiple No Wait (+400) – Attempts to lock multiple records in the same file and returns control if the records are already locked.

Browse
Available with the Open operation, this button allows you to choose a file to open. The file name you select is copied to the Key Buffer.

To Do
Allows you to perform the same operation a specified number of times. This option is helpful when you want to perform repetitive operations. For example, if you want to insert the same record 100 times, select the Insert operation and specify 100 in this field.

Done
Contains the number of times the MicroKernel has executed the operation.

Key

Number
For most Get operations, specifies a key number, or index path, to follow for the current operation. For other operations, specifies such information as file open mode, encryption, or logical disk drive. This control corresponds with the Key Number parameter.

Buffer
Specify the path for the data file for which you want to perform a Btrieve operation.

Position
Indicates the position of the cursor within the Key Buffer.

Data

Length
Specifies the length (in bytes) of the Data Buffer. The default is 100. For every operation that requires a data buffer, you must specify a buffer length. On many operations, the MicroKernel returns a value to the Data Length. Generally, you should always specify a Data Length before you execute an operation. This

control corresponds with the Data Buffer Length parameter.

Buffer Specifies a data value. For read and write operations, the Data Buffer contains records. For other operations, the Data Buffer contains file specifications, filtering conditions, and other information the MicroKernel needs for processing the operation. This control corresponds with the Data Buffer parameter.

Position Indicates the position of the cursor within the Key or Data Buffer.

Edit Key Displays a dialog that allows you to edit the Key Buffer using ASCII or hexadecimal values. For more information, refer to “Editing the Key or Data Buffer (OS/2 Clients only)”.

Edit Data Displays a dialog that allows you to edit the Data Buffer using ASCII or hexadecimal values. For more information, refer to “Editing the Key or Data Buffer (OS/2 Clients only)”.

Status Displays a numeric status code returned by the MicroKernel and a brief message explaining the result of a Btrieve operation. For detailed information about these status codes and messages, refer to the Status Codes and Messages manual.

Status Help Provides help for all status codes.

Execute Performs the currently specified operation.

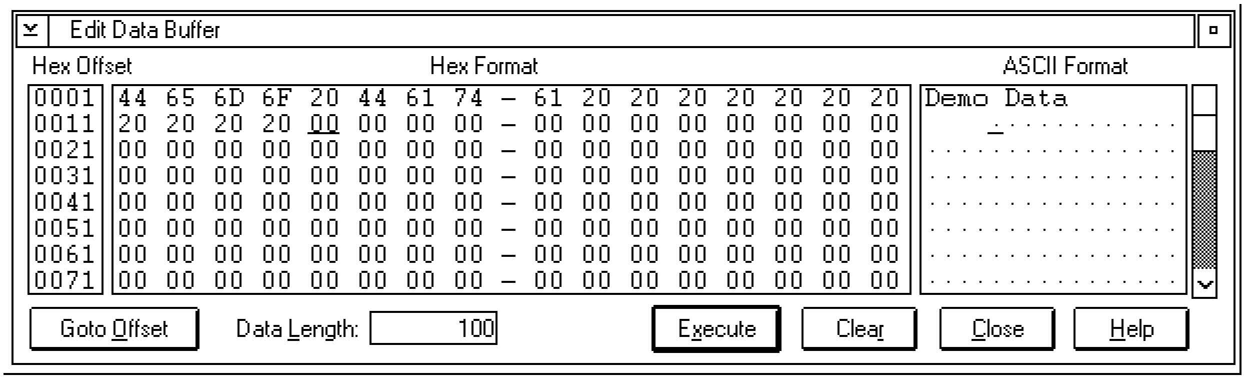
Editing the Key or Data Buffer (OS/2 Clients only)

In the OS/2 version of the Function Executor, you can edit the contents of the Key and Data buffers from the utility's main window. However, the Function Executor offers separate dialog boxes that provide more information about the buffers and allow you to perform more extensive editing. If you have multiple files open, these dialog boxes reflect the contents of the buffers for the current file. As you switch among open files, the contents of these dialog boxes change.

To use the other dialog boxes in the Function Executor utility:

- Click Edit Key or Edit Data on the main window. Figure 6-3 shows the Edit Data Buffer dialog box.

Figure 6-3
Edit Data Buffer Dialog



```
0001  44 65 6D 6F 20 44 61 74 – 61 20 20 20 20 20 20 20   Demo Data
0011  20 20 20 20 00 00 00 00 – 00 00 00 00 00 00 00 00       .............
0021  00 00 00 00 00 00 00 00 – 00 00 00 00 00 00 00 00   ................
0031  00 00 00 00 00 00 00 00 – 00 00 00 00 00 00 00 00   ................
0041  00 00 00 00 00 00 00 00 – 00 00 00 00 00 00 00 00   ................
0051  00 00 00 00 00 00 00 00 – 00 00 00 00 00 00 00 00   ................
0061  00 00 00 00 00 00 00 00 – 00 00 00 00 00 00 00 00   ................
0071  00 00 00 00 00 00 00 00 – 00 00 00 00 00 00 00 00   ................

Data Length: 100
```

Table 6-3 lists the controls in this dialog box.

Table 6-3
Edit Data Buffer Dialog Controls

Control	Description
Hex Offset	Displays the position of the first byte in the row, relative to the beginning of the buffer (Position 1).
Hex Format	Displays the contents of the Key or Data buffer in hexadecimal values.
ASCII Format	Displays the contents of the Key or Data buffer in ASCII values.
Go To Offset	Allows you to specify an offset value in the buffer and go to that position quickly.
Data Length	Specifies the length of the Data buffer.
Execute	Executes the current operation.
Clear	Clears the contents of the buffer.
Close	Closes the dialog.
Help	Provides help for the dialog.

Performing Operations

To perform an operation, specify values for the appropriate controls and click Execute. Because Btrieve provides many operations, this chapter cannot explain them all. The following sections discuss some common operations.

Opening a File

You can open only Pervasive.SQL data files using the Function Executor.

To open a data file:

1. Use the List box to select the Open (0) operation or enter 0 in the Current box.
2. Enter the path of a data file in the Key buffer.
3. Click Execute or press Enter.

Detecting the Presence of a Specific Key Value

To detect the presence of a specific key value in a data file:

1. Open the data file or select it from the list of open files.
2. Specify the Get Equal (5) operation with a Get Key (+50) bias.

   You can use one of the following methods to specify this operation:

   - Use the List box to select the Get Equal (5) operation and select the Get Key Bias (+50) check box.
   - Enter 55 in the Current box.
3. In the Key Number box, enter the number of the key you want to verify.
4. In the Key Buffer, enter the key value you want to detect.
5. Click Execute or press Enter.

Manipulating Btrieve Data Files with the Maintenance Utility

This chapter discusses the following topics:

- “Maintenance Utility Overview”
- “Btrieve Interactive Maintenance Utility”
- “Btrieve Command-Line Maintenance Utility (BUTIL)”

Maintenance Utility Overview

Pervasive.SQL provides both an interactive Maintenance utility and a command-line Maintenance utility. Both Maintenance utilities perform the following common file and data manipulations:

- Create new data files based on file and key specifications you define.
- Provide file and key specifications for existing data files.
- Set and clear owner names for data files.
- Create and drop indexes on data files.
- Import and export ASCII sequential data.
- Copy data between Pervasive.SQL data files.
- Recover changes made to a file between the time of the last backup and a system failure.

While both utilities provide the same core functionality, minor differences exist. For example, the interactive Maintenance utility allows you to create description files based on file and key specifications you define. The command-line Maintenance utility allows you to start and stop continuous operation on a file or set of files locally on the server.

Before you use either Maintenance utility, you should be familiar with Btrieve fundamentals, such as files, records, keys, and segments. For information about these topics, refer to the *Pervasive.SQL Programmer's* Guide.

Btrieve Interactive Maintenance Utility

The Interactive Maintenance utility runs on Win16 (client/server only), Win32 (Pervasive.SQL client/server and workstation products), and OS/2 (client/server only) platforms. Use this utility if you prefer a graphical interface or if you want to create a description file. This section provides the following information:

- "Starting the Btrieve Maintenance Utility"
- "Extended File Support"
- "File Information Editor Overview"
- "Loading Information From an Existing File"
- "Creating a New File"
- "Compacting Btrieve Data Files"
- "Showing and Hiding 6.x Data"
- "Specifying a Key's Alternate Collating Sequence"
- "Generating a Statistics Report"
- "Setting and Clearing Owner Names"
- "Creating and Dropping Indexes"
- "Importing, Exporting, and Copying Records"
- "Recovering Data After a System Failure"

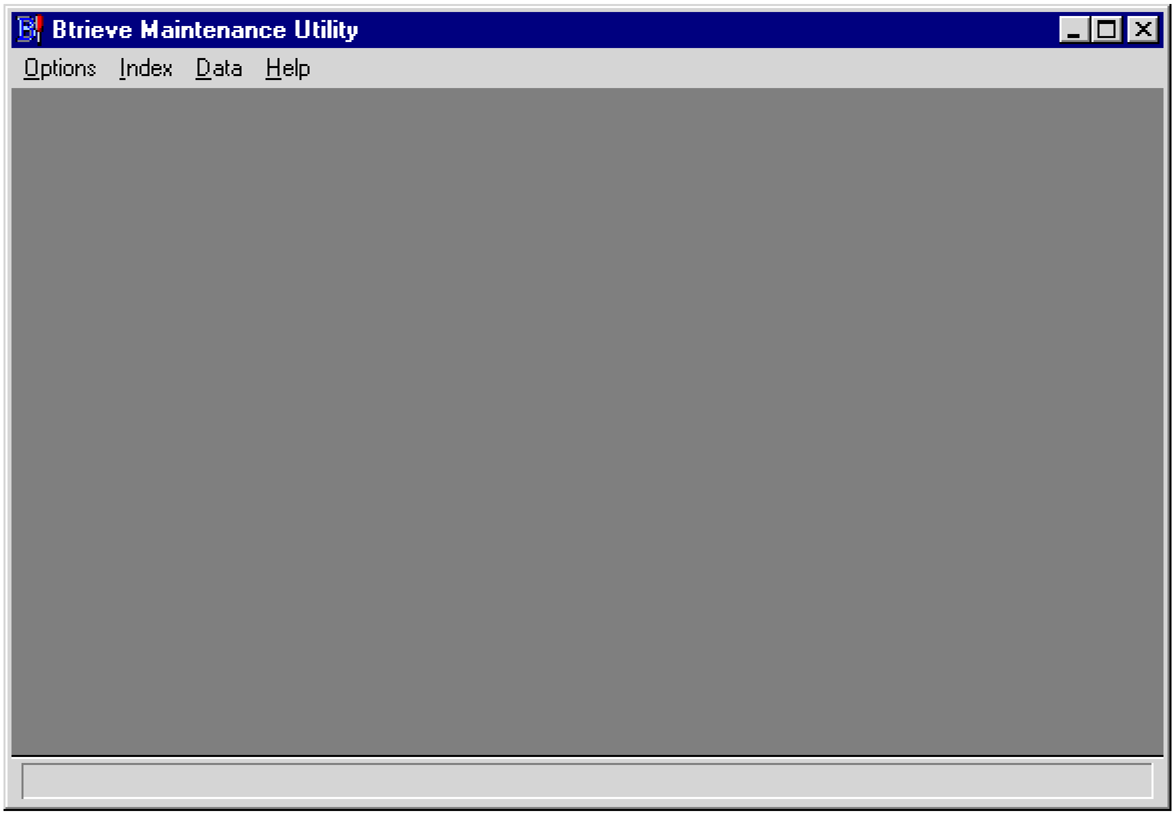
Starting the Btrieve Maintenance Utility

To start the Btrieve Maintenance utility:

- From the **Start** menu, select **Programs** and then **Pervasive SQL 7 Server** , and then click **Maintenance (Win16)** or **Maintenance (Win32)** .

  The Btrieve Maintenance utility's main window displays as illustrated in Figure 7-1.

Figure 7-1
Btrieve Maintenance Utility Main Window



Menu Options

The interactive Maintenance utility provides the following menus:

Options — Allows you to display the File Information Editor, set and clear owner names, generate statistics reports, and exit the utility.

Index — Allows you to create and drop indexes.

Data — Allows you to load data from ASCII files, save data to ASCII files, copy records between data files, and perform a roll forward operation to recover changes made to a data file between the time of the last backup and a system failure.

Help — Provides access to the Maintenance utility help system.

Getting Help

To access the Maintenance utility help system, choose a command from the **Help** menu, as follows:

Getting Help — Explains how to use the Maintenance utility help system.

Index — Provides a list of Maintenance utility help topics.

Help on Help — Explains how to use the help system.

About — Displays copyright information and the version number. It also provides the version number of the MicroKernel Database Engine and Btrieve client Requester, if they are loaded.

In addition, you can display help for a particular dialog box by clicking the **Help** button contained in that dialog box.

Extended File Support

The size of the MicroKernel data file can be larger than the operating system file size limit. When you export data from an extended MicroKernel file to an unformatted file, the size of the unformatted file can exceed the MicroKernel file size limit because of the differences in the physical format.

The Interactive Maintenance utility detects that the unformatted file has exceeded the file size limit (2 GB) and starts creating extension files. This process is transparent. Extension files and the original unformatted file must reside on the same volume. The extension file uses a naming scheme similar to the MicroKernel Database Engine. The first extension file is the same base file name with '.~01' extension. The second extension file is '.~02,' and so on. These numbers are printed in hex. The maximum number of extension files is 255; the 255th extension file (the last extension file) has an extension of '.~ff'.

Additionally, when you import data from an unformatted file, the utility detects if the file has extensions and loads the data from the extension file.

File Information Editor Overview

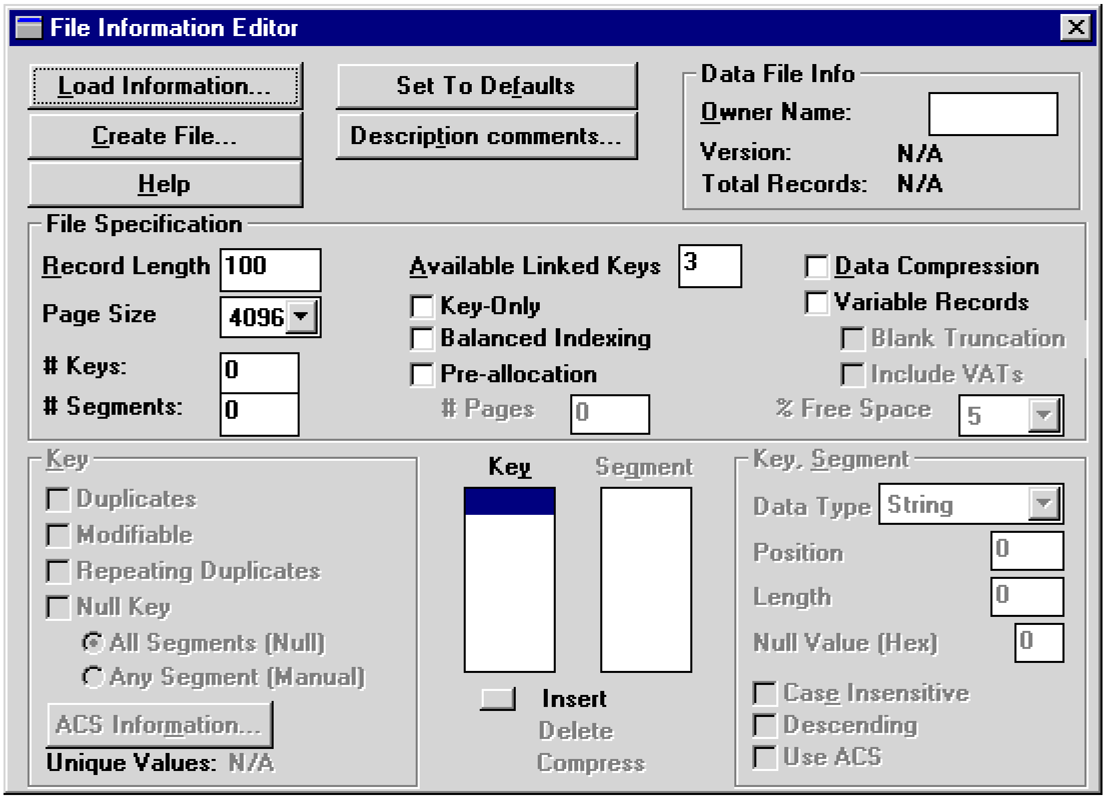
This section provides general information about the File Information Editor dialog where you can create new files based on file and key specifications you construct. Because this dialog box allows you to load information based on an existing file, it is also useful for viewing file and key specifications on existing data files. You can also create a new file based on the file and key specifications of an existing file (similar to "CLONE" in the command-line Maintenance utility).

To open the File Information Editor dialog:

1. Start the Btrieve Maintenance utility.
2. From the utility's main window, select **Show Information Editor** from the **Options** menu.

   The File Information Editor (Figure 7-2) opens.

Figure 7-2
File Information Editor



Owner Name:
Version: N/A
Total Records: N/A

Record Length 100
Page Size 4096
Keys: 0
Segments: 0
Available Linked Keys 3
□ Key-Only
□ Balanced Indexing
□ Pre-allocation
Pages 0
□ Data Compression
□ Variable Records
□ Blank Truncation
□ Include VATs
% Free Space 5

□ Duplicates
□ Modifiable
□ Repeating Duplicates
□ Null Key
● All Segments (Null)
○ Any Segment (Manual)
Unique Values: N/A

Data Type String
Position 0
Length 0
Null Value (Hex) 0
□ Case Insensitive
□ Descending
□ Use ACS

File Information Editor Dialog Elements

At the top of the dialog box, the following buttons appear:

Load Information
Loads information based on an existing file. When you load information, you are not editing the existing file. Instead, you are loading a copy of the information about that file.

Create File
Creates a new file based on current information in the dialog box.

Set to Defaults
Sets the controls to default values.

Description Comments
If you are creating a description file, allows you to enter notes about the file.

Show 6.x Attributes
Displays controls specific to 6.x and later files, which are unavailable when you load information based on a pre-6.0 file. (This button is unavailable unless you load information based on a pre-6.x file.)

Help
Displays help for the File Information Editor dialog.

The **Data File Info** box, also at the top of the File Information Editor, contains the following controls:

Owner Name
Provides a text box you can use to specify the owner name, if applicable, for an existing file.

Version
Earliest version of the MicroKernel that can read all the attributes of the file. For example, if you created a file using the 6.15 MicroKernel but did not use any attributes specific to 6.15, the Maintenance utility

displays 6.0 as the version number.

Total Records Total number of records in the file.

The **File Specification** box is in the middle of the File Information Editor. Table 7-1 describes the controls in this box.

Table 7-1
File Specification Controls

Control	Description	Range	Default
Record Length	Specifies the logical data record length (in bytes) of the fixed-length records in a file.	4–4,088	100
Page Size	Specifies the physical page size (in bytes) for the file.	512–4,096	4,096
# Keys	Indicates the number of distinct keys (as opposed to key segments) currently defined in the Editor. Reflects the number of keys in the Key list.	0–119	0
# Segments	Indicates the number of key segments currently defined in the Editor. Reflects the number of segments in the Segment list.	0–119	0
Available Linked Keys	Specifies how many 8-byte place holders you want to reserve for future linked-duplicatable keys. If you are loading information based on an existing data file, this value reflects the number of place holders currently available in that file. (The number of originally reserved place holders is not stored in the file.)	0–119	3
Key-Only	Indicates whether the file is key-only. Not applicable if you turn Data Compression on, if you turn Variable Records on, or if you define more than one key for the file.	On or Off	Off
Balanced Indexing	Specifies that the file uses the balanced indexing method of managing key pages.	On or Off	Off
Pre-allocation	Specifies that the file uses preallocated pages.	On or Off	Off
# Pages	Specifies the number of pages you want preallocated when you create the file. Applicable only if Pre-allocation is turned on. If you are loading information based on an existing data file, this value reflects the number of unused, preallocated pages left in that file. (The number of originally preallocated pages is not stored in the file.)	1–65,535	0
Data Compression	Specifies that the file uses data compression. Not applicable for key-only files or files that use blank truncation.	On or Off	Off
Variable Records	Specifies that the file can contain variable-length records.	On or Off	Off
Blank Truncation	Specifies whether the file uses blank truncation on variable records to conserve disk space.	On or Off	Off

	Applicable only if Variable Records is turned on.		
Include VATs	Specifies whether the file supports Variable-tail Allocation Tables for faster access to data in very long records. Applicable only if Variable Records is turned on.	On or Off	Off
% Free Space	Specifies the amount of unused space a file's variable pages must have available before the MicroKernel creates a new variable page. Applicable only if Data Compression or Variable Records are turned on.	5, 10, 20, or 30	5

At the bottom middle of the dialog box, the **Key** list shows the key numbers defined in a file. (For 6.x and later files, these key numbers do not have to be consecutive; they can have gaps between them.) The Maintenance utility displays the highlighted key's specifications in the **Key** box at the bottom left of the dialog box.

Also at the bottom middle of the dialog box, the **Segment** list shows the key segment numbers defined for the key highlighted in the **Key** list. The Maintenance utility displays the highlighted segment's specifications in the **Segment** box at the bottom right of the dialog box.

In addition, the following buttons appear under the Key and Segment lists:

Insert	Defines a new key or segment.
Delete	Removes the highlighted key or segment specification.
Compress	Renumbers the keys consecutively. You can use this button to remove gaps that result from deleting a key specification.

Note: Because these buttons control key specifications for a file you want to create, you cannot use them to operate on keys in an existing file. If you want to create or drop an index on an existing file, refer to "Creating and Dropping Indexes".

At the bottom left in the dialog box is the **Key** group box. Table 7-2 describes the controls in this area. These controls are specific to the designated key (that is, the key highlighted in the **Key** list), not just to the current key segment. When you change the setting for one of these controls, the change affects *all* segments of the specified key.

Table 7-2
Key Specification Controls

Control	Description	Default
Duplicates	Specifies that the key can have duplicate values.	On
Modifiable	Specifies that the key value can be modified after creation.	On
Repeating Duplicates	Specifies that the MicroKernel uses the repeating-duplicatable method of storing duplicate key values.	Off
Null Key	Specifies that the key has a null value.	Off
All Segments	Specifies that if all key segments in the record contain the null value,	Off

(Null)	the MicroKernel does not include that record in the key path. Applicable only if Null Key is turned on.	
Any Segment (Manual)	Specifies that if one or more key segments contain the null value, the MicroKernel does not include that record in the key path. Applicable only if Null Key is turned on.	Off
ACS Information	Allows you to specify an alternate collating sequence (ACS) for the key. Applicable only if the **Use ACS** check box is selected for a segment of the key.	Off
Unique Values	Indicates the number of unique key values in the file. Applicable only if you are loading information based on an existing data file.	N/A

At the bottom right in the dialog box is the Key Segment group box. Table 7-3 describes the controls in this area. These controls are specific to the designated key segment (that is, the segment highlighted in the **Segment** list),

Table 7-3
Key Segment Specification Controls

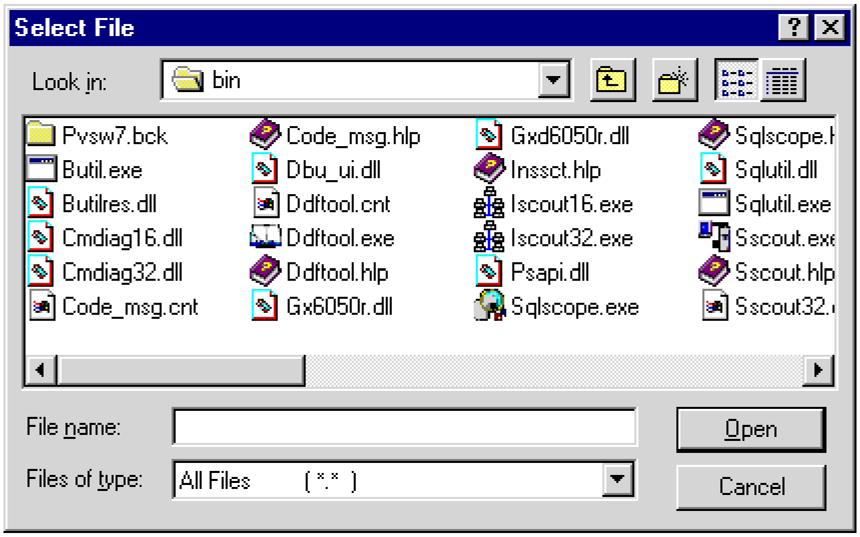
Control	Description	Default
Data Type	Specifies a data type for the key segment.	string
Position	Specifies by number the relative starting position of the beginning of this key segment in the record. The value cannot exceed the record length.	1
Length	Specifies the length (in bytes) of the key segment. This value cannot exceed the limit dictated by the data type for the segment. The total of key position and key length cannot exceed the record length.	10
Null Value	Specifies the null character value (in hexadecimal) for the key segment. Applicable only if the Null Key check box is selected for the key.	Binary zero
Case Insensitive	Specifies whether the segment is sensitive to case. Applicable only for STRING, LSTRING, and ZSTRING data types or for keys that do not use an ACS.	On
Descending	Specifies that the MicroKernel sort the key segment values in descending order (that is, from highest to lowest).	Off
Use ACS	Specifies that the segment uses the alternate collating sequence defined for the key. Applicable only for string, lstring and zstring data types that are case sensitive.	Off

Loading Information From an Existing File

To load information from an existing data file into the File Information Editor:

1. Click **Load Information** at the top of the **File Information Editor** dialog box. The **Select File** dialog box (Figure 7-3) appears.

Figure 7-3
Select File Dialog



2. Specify the name and path of the file for which you want to load information. (By default, data files have a .mkd extension.)

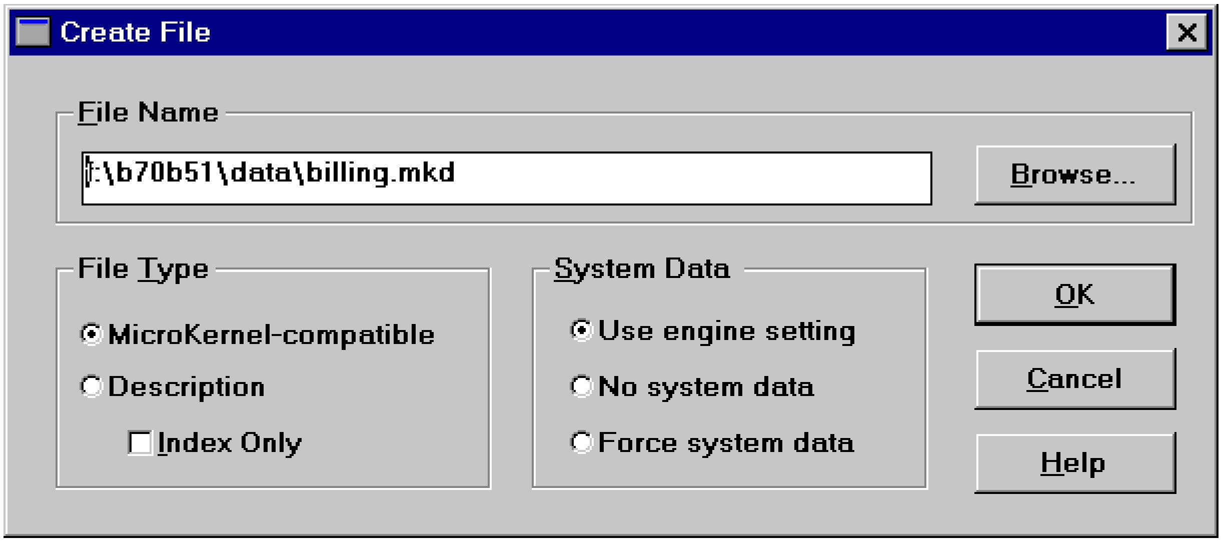
   The Maintenance utility first attempts to open the specified file as a data file. If the file requires an owner name, the utility prompts you for one. (Because owner names are optional, the file you open may not require an owner name.) If the specified file is not a data file, the utility then attempts to open the file as a description file.

Creating a New File

To create a new file based on the current information in the File Information Editor:

1. Click **Create File** at the top of the **File Information Editor** dialog box. The **Create File** dialog box (Figure 7-4) appears.

Figure 7-4
Create File Dialog



File Name: f:\b70b51\data\billing.mkd

File Type: ● MicroKernel-compatible ○ Description □ Index Only

System Data: ● Use engine setting ○ No system data ○ Force system data

2. Specify the controls in the Create File dialog, which are described in Table 7-4.

Table 7-4
Create File Dialog Controls

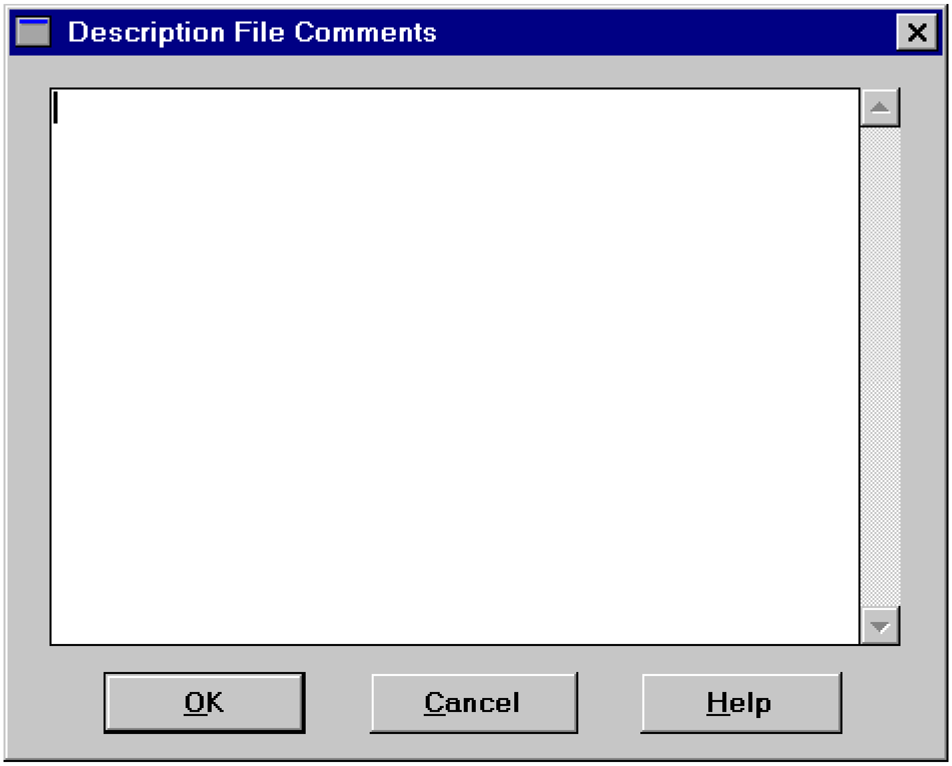
Control	Description	Default
File Name	Specifies a name and path for the file. By default, data files have a .mkd extension.	N/A
File Type	Specifies the type of file to create. If you are creating a description file, you can use the Index Only option, which creates a description file you can use with the BUTIL utility to add an index to an existing data file. (For more information, refer to "Creating Indexes".)	MicroKernel-compatible
System Data	Determines whether the utility includes system data in the file. If you choose Use Engine Setting, the utility uses the setting for the "System Data" configuration option. If you choose No System Data, the utility does not create system data, regardless of the engine configuration. If you choose Force System Data, the utility creates system data, regardless of the engine configuration.This is applicable only if the file type is MicroKernel-compatible.	Use Engine Setting

Adding Comments to a Description File

To add comments to a description file you are creating:

1. Click **Description Comments** . The Description File Comments dialog box (Figure 7-5) appears.

Figure 7-5
Description File Comments Dialog



2. Enter a block of comments up to 5,120 characters long.

3. Click **OK** when you are finished entering comments.

Compacting Btrieve Data Files

This section describes how to remove unused space in a Btrieve data file, which ultimately decreases the file's size. You can also perform this procedure using the command-line Maintenance utility (BUTIL).

To compact a Btrieve file:

1. Start the Maintenance utility.

2. Select **Show Information Editor** from the **Options** menu.

3. Choose **Load Information** and select the file you want to compact.

4. Select **Create File** , give the file a new name (which creates a clone) in the Create File dialog, and click **OK** .

5. From the **Data** menu on the main window, select **Save** . In the Save Data dialog, enter the name of the original file in the **From MicroKernel File** box and then specify a name for the output file (for example, <original file>.out) in the **To Sequential File** box.

6. Click **Execute** . The Save Data dialog displays the results of the save. Click **Close** .

7. From the **Data** menu, select **Load** . In the Load Data dialog, enter the name of the sequential data file you just saved in the **From Sequential File** box. Then enter the name of the clone file you created in Step 4 in the **To MicroKernel File** box.

8. Click **Execute** . The Loading Data dialog displays the results of the load. Click **Close** .

   You can now compare the size of the original file to the clone file to verify the reduction in size.

Showing and Hiding 6.x Data

When you load a pre-6.0 file, all the controls in the File Information Editor that are specific to 6.x and later are unavailable unless you click the **Show 6.** x **Data** button. For example, one feature that is specific to 6.x and later is the use of variable-tail allocation tables, or VATs.

This button is useful for users working in environments that use both pre-6.0 and 6.x and later MicroKernels. By hiding the 6.x-specific controls, you can avoid unintentionally creating a 6.x file. (Pre-6.0 MicroKernels cannot access 6.x files.)

To display the 6.x-specific controls:

- Click **Show 6.** x **Data** .

  The Show 6.*x* Data button is unavailable unless you are working with pre-6.0 files.

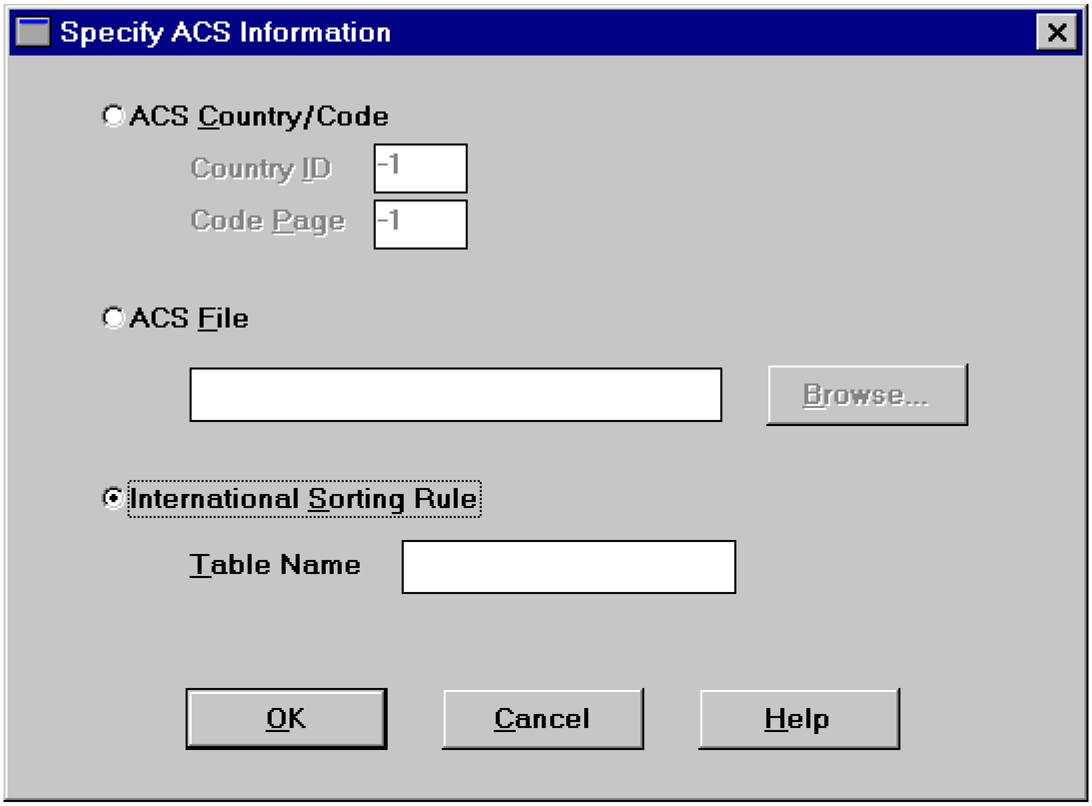
Specifying a Key’s Alternate Collating Sequence

To specify a key’s alternate collating sequence:

1. Click **ACS Information** .

   The Maintenance utility displays the **Specify ACS Information** dialog box (Figure 7-6).

Figure 7-6
Specify ACS Information Dialog



○ ACS Country/Code

Country ID: -1

Code Page: -1

○ ACS File

● International Sorting Rule

Table Name:

2. You can specify either a country ID and code page, an ACS file name, or an International Sorting Rule (ISR) as follows:

Table 7-5
ACS Information Controls

Control	Description	Default
ACS Country/Code		
Country ID	An Intel-format number that identifies your country. Refer to your operating system's documentation for specific information.	-1
Code Page	An Intel-format number that identifies the code page you want to use. Refer to your operating system's documentation for specific information.	-1
ACS File	Specifies the fully qualified file name of the alternate collating sequence file.	N/A
International Sorting Rule	When you click this radio button you can specify a specific ISR table for sorting international data. Pervasive.SQL 7 provides a set of pre-generated ISR tables,**1 which are listed in the *Pervasive.SQL Programmer's Guide* .	

**1 Subsequent releases of Pervasive.SQL will include more ISR tables for the various languages supported.

3. When you specify a country ID and code page ID, the MicroKernel stores the locale-specific collating sequence in the data file. Moreover, the MicroKernel can insert new key values correctly, even if the locale changes.

4. When you specify an ACS file name for a data file, the MicroKernel copies the contents of the ACS file into

the data file. (That is, the data file does not contain the file name of the ACS file.) The ACS identifies itself using an eight-digit name (such as UPPER). Subsequently, when you view the ACS information for a data file, the Maintenance utility displays this eight-digit name, not the file name of the original ACS.

5. When you specify an ACS file name for a description file, the Maintenance utility copies the actual path and file name of the ACS file into the description file. Subsequently, when you view the ACS information for a description file, the Maintenance utility attempts to locate the specified ACS file.

   To specify an ACS that sorts string values using an ISO-defined, language-specific collating sequence, you must specify an ISR table name. The Table Name field is limited to 16 characters. For more information on ISRs, refer to the *Pervasive.SQL Programmer's Guide* .

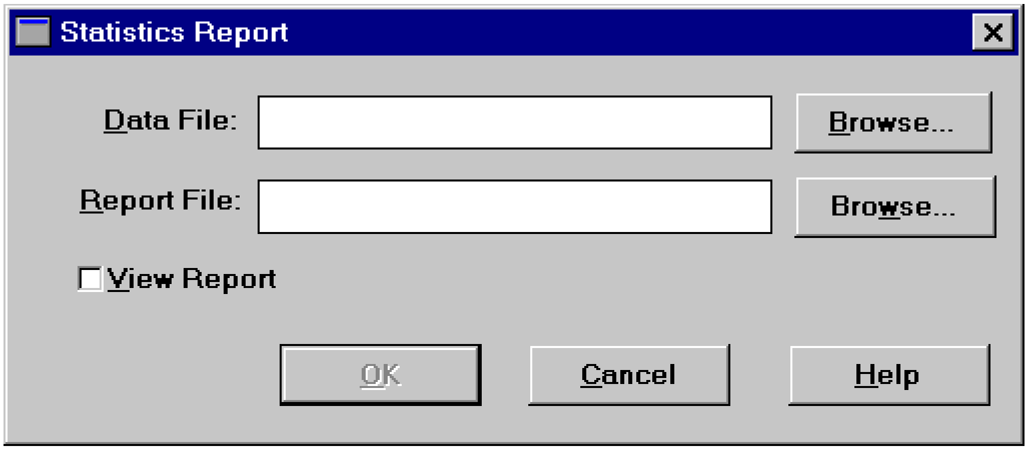
Generating a Statistics Report

Generating a statistics report is a good way to determine whether a file can be logged by the MicroKernel's transaction durability feature.

To examine a statistics report for an existing data file:

1. Choose **Create Stat Report** from the **Options** menu on the main window. The Maintenance utility displays the **Statistics Report** dialog box (Figure 7-7).

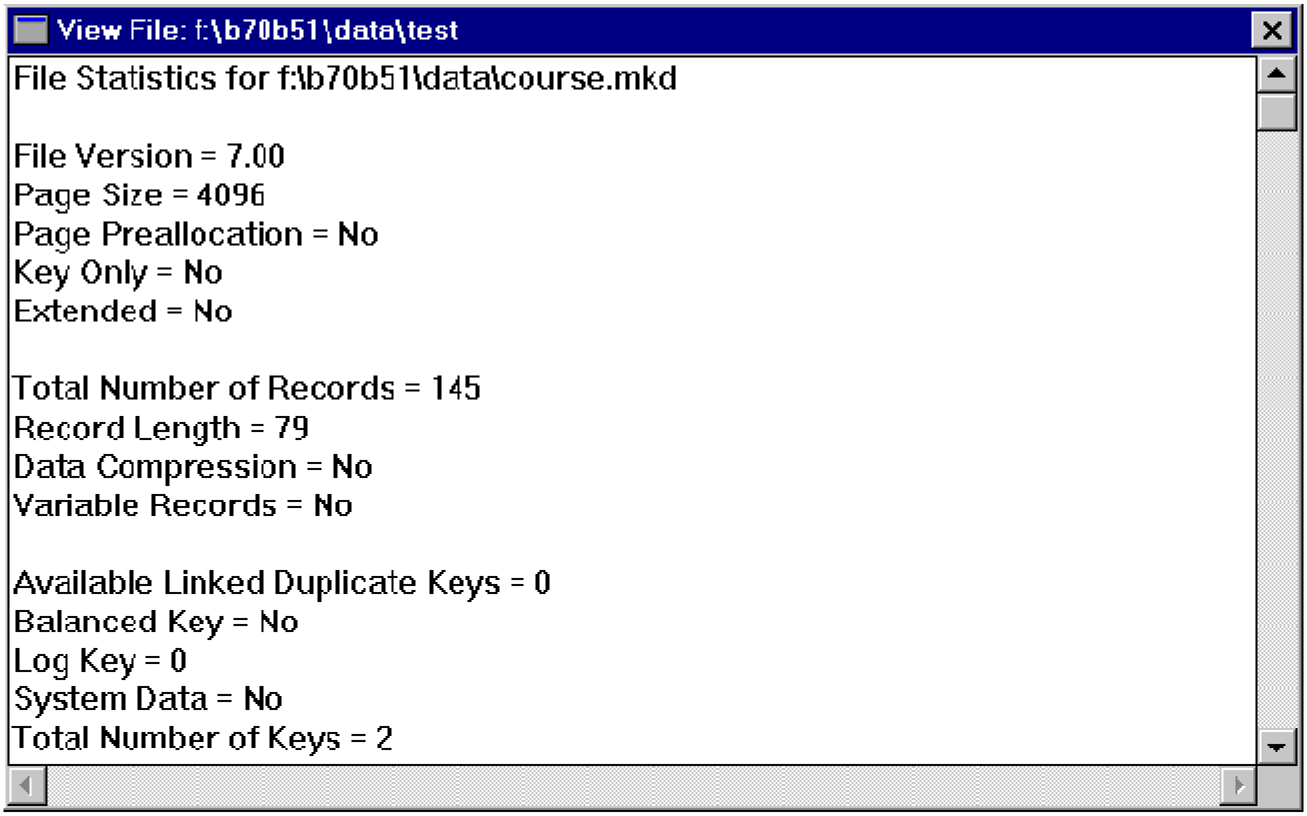
Figure 7-7
Statistics Report Dialog



2. Specify a data file to use and a report file name. If you want to view the report when it is created, select the **View Report** check box.

   If you choose to view the report, the Maintenance utility displays the View File window as shown in Figure 7-8.

Figure 7-8
Statistics Report Example

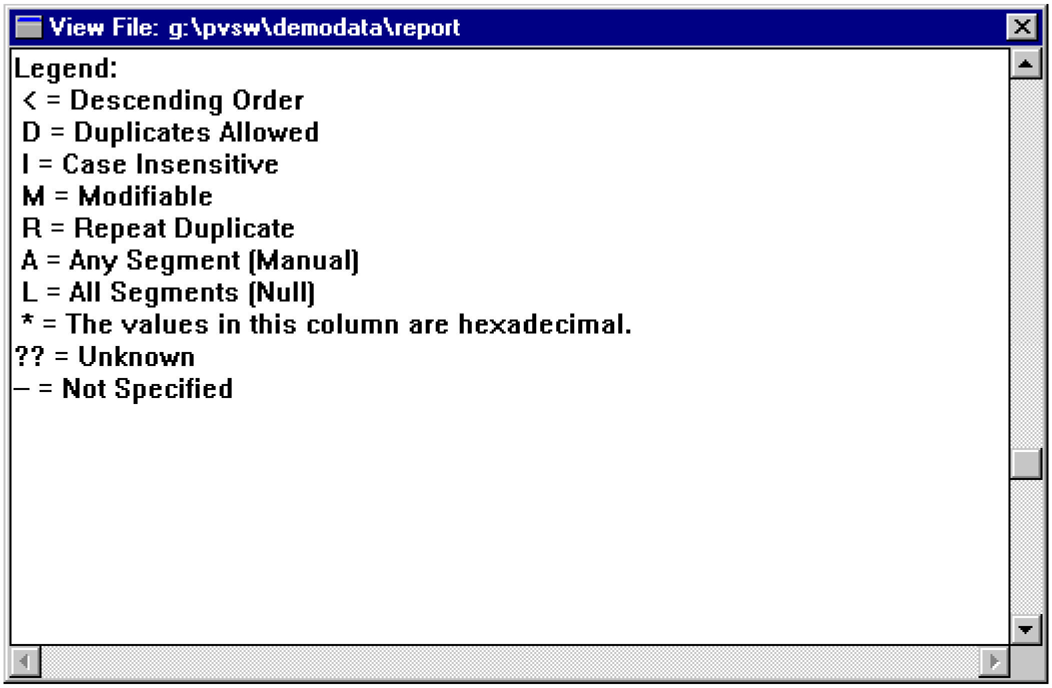


```
File Statistics for f:\b70b51\data\course.mkd

File Version = 7.00
Page Size = 4096
Page Preallocation = No
Key Only = No
Extended = No

Total Number of Records = 145
Record Length = 79
Data Compression = No
Variable Records = No

Available Linked Duplicate Keys = 0
Balanced Key = No
Log Key = 0
System Data = No
Total Number of Keys = 2
```

The informational headings in a status report correspond to the controls in the File Information Editor, which is described in the "File Information Editor Overview".

The legend at the bottom of the statistics report, shown in Figure 7-9, explains the symbols used in the key/segment portion of the report. This information includes items such as the number of keys and key segments, the position of the key in the file, and the length of the key.

Figure 7-9
Statistics Report Key/Segment Information and Legend



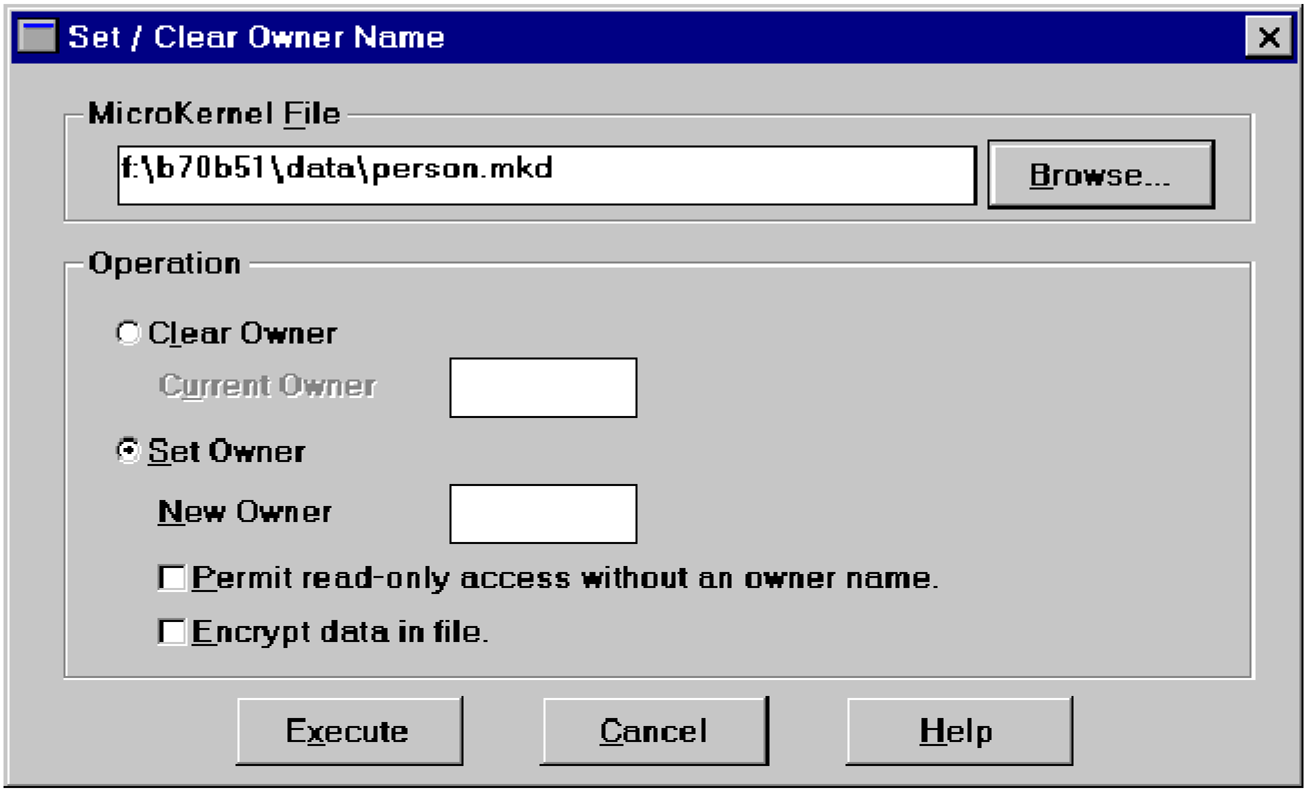
```
Legend:
 < = Descending Order
 D = Duplicates Allowed
 I = Case Insensitive
 M = Modifiable
 R = Repeat Duplicate
 A = Any Segment (Manual)
 L = All Segments (Null)
 * = The values in this column are hexadecimal.
?? = Unknown
-- = Not Specified
```

Setting and Clearing Owner Names

To set or clear an owner name:

1. Choose **Set/Clear Owner** from the Options menu. The **Set/Clear Owner Name** dialog box (Figure 7-10) appears.

Figure 7-10
Set/Clear Owner Name Dialog



Set / Clear Owner Name

MicroKernel File: f:\b70b51\data\person.mkd

Operation

○ Clear Owner

Current Owner:

● Set Owner

New Owner:

□ Permit read-only access without an owner name.

□ Encrypt data in file.

2. In the **MicroKernel File** box, specify the file for which you want to set or clear an owner name. Then, to clear the owner name, click **Clear Owner** and specify the file's owner name in the **Current Owner** box.

3. To set the owner name, click **Set Owner** and specify the file's new owner name in the **New Owner** box. Select the **Permit read-only access without an owner name** checkbox to allow anyone to read-only access to the records.

   Select the **Encrypt data in file** checkbox to ensure that unauthorized users do not examine your data using a debugger or a file dump utility. Only select this option if data security is important to your environment as this requires additional processing time.

Creating and Dropping Indexes

An *index* is a structure that sorts all the key values for a specific key. You can use the Maintenance utility to create or drop indexes on a data file.

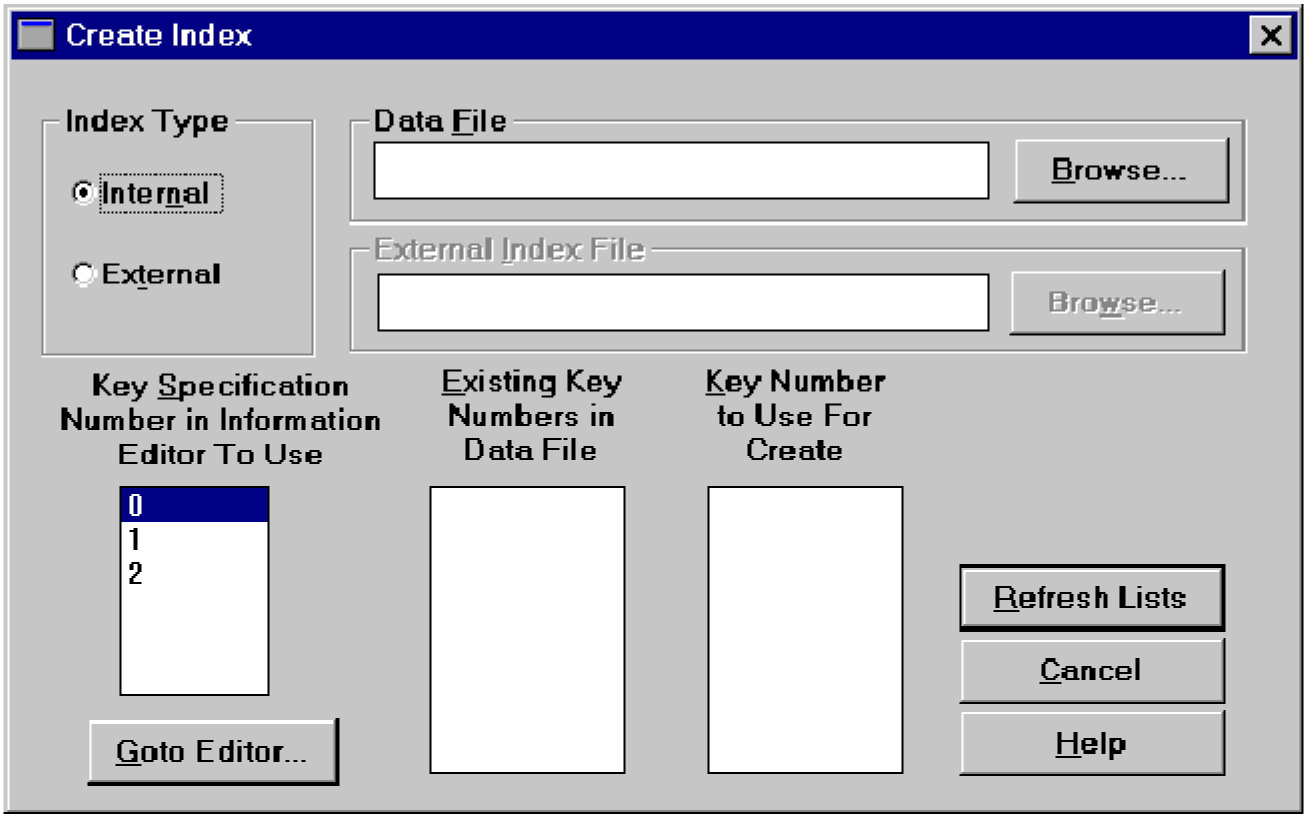
Creating Indexes

You cannot create an index for a file unless the file has at least one key defined.

To create an index:

1. Choose **Create** from the **Index** menu, which opens the **Create Index** dialog box (Figure 7-11).

Figure 7-11
Create Index Dialog



2. Complete the following data boxes in the **Create Index** dialog box.

Index Type

Specify whether to create an internal or external index. Internal indexes are dynamically maintained as part of the data file. External indexes are separate files you generate as needed.An external index file is a standard data file that contains records sorted by the key you specify. Each record consists of the following:#• A 4-byte address identifying the physical position of the record in the original data file

- A key value

Data File

Specify the name of the data file for which you want to create the index.

External Index File

Specify the name of the file to generate for an external index. Not applicable for internal indexes.

Key Specification Number in Information Editor to Use

Lists the key numbers defined in the File Information Editor. If the file contains a system-defined log key (also called system data) but the key has been dropped, this list includes SYSKEY, which you can select to re-add the system-defined log key to the file.

Existing Key Numbers in Data File

Click **Refresh Lists** to display the key number defined for the file. If the file contains a system-defined log key, this list includes SYSKEY.

Key Number to Use For Create

Click **Refresh Lists** to display the key numbers available (that is, not defined for the file). Highlight the key number you want to use when creating the index.

3. You can click **Go To Editor** to display the **File Information Editor** dialog box, which shows more complete information about the key. You can click **Refresh Lists** to read key information from the data file and refresh the **Existing Key Numbers in Data File** and **Key Number to Use For Create** lists. You must click **Refresh Lists** before you can create an index.

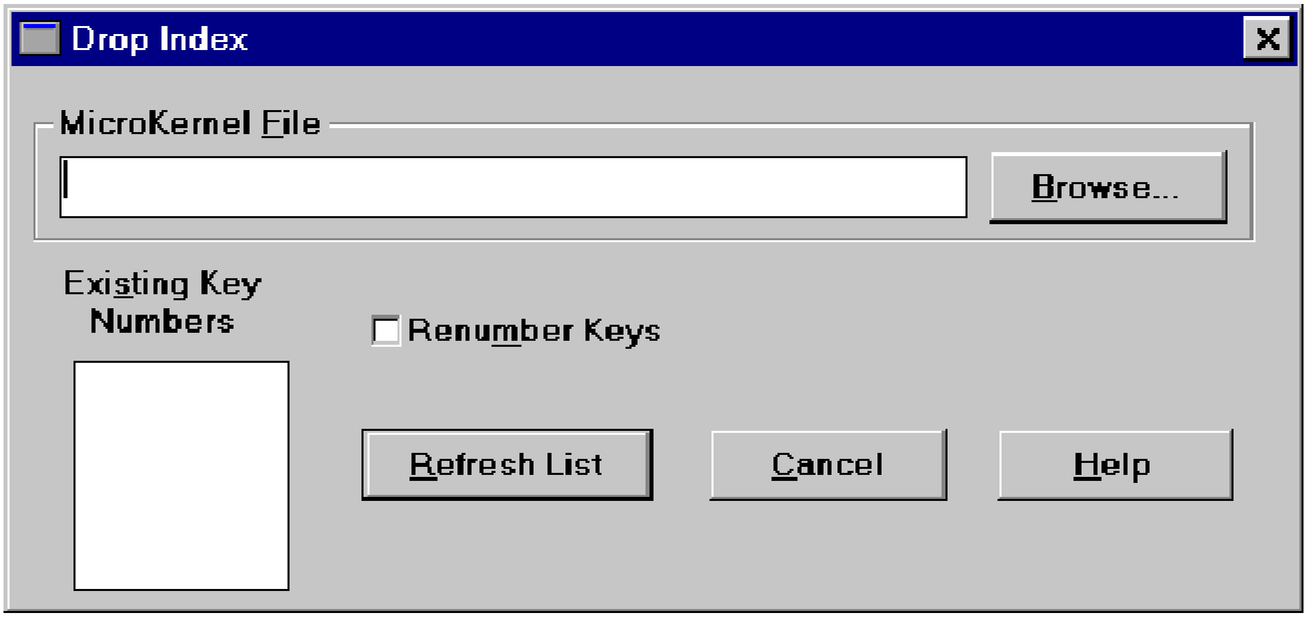
4. When you have completed the **Create Index** dialog box, click **Execute** to create the index. The amount of time required to create the index depends on how much data the file contains.

Dropping Indexes

To drop an index:

1. Choose **Drop** from the **Index** menu. The **Drop Index** dialog box (Figure 7-12) appears.

Figure 7-12
Drop Index Dialog



2. Complete the following data boxes in the **Drop Index** dialog box.

MicroKernel File — Specify the name of the data file from which you want to drop the index.

Existing Key Numbers — Click **Refresh List** to display the key number defined for the file. Highlight the number of the key whose index you want to drop. If the file contains a system-defined log key, this list includes SYSKEY, which you can select to drop the system-defined log key from the file.

Renumber Keys — Renumbers the keys consecutively. Click this button to remove gaps that result from deleting an index.

3. Click **Refresh List** to get the key information from the file you have specified.

Importing, Exporting, and Copying Records

The Load, Save, and Copy commands in the **Data** menu allow you to import, export, and copy records in data files. You can also recover data after a system failure with the Roll Forward feature.

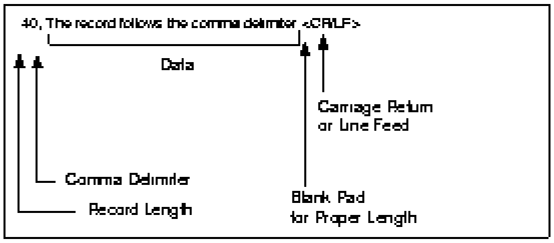
Importing and Exporting ASCII File Format

When you save data, records in the ASCII file have the following format. You can use an ASCII text editor to create files that you can load, as long as they adhere to these specifications. Note that most text editors do not support editing binary data.

- The first field is a left-adjusted integer (in ASCII) that specifies the length of the record. (When calculating this value, ignore the carriage return/line feed that terminates each line.) The value in this first field matches the record length specified in the data file.
    - For files with fixed-length records, the length you specify should equal the record length of the data file.
    - For files with variable-length records, the length you specify must be at least as long as the fixed-record length of the data file.
- A separator (a comma or a blank) follows the length field.
- The record data follows the separator. The length of the data is the exact number of bytes specified by the length field. If you are creating an import ASCII file using a text editor, pad each record with blank spaces as necessary to fill the record to the appropriate length.
- A carriage return/line feed (0D0A hexadecimal) terminates each line. The Maintenance utility does not insert the carriage return/line feed into the data file.
- The last line in the file must be the end-of-file character (CTRL+Z or 1A hexadecimal). Most text editors automatically insert this character at the end of a file.

Figure 7-13 shows the correct format for records in the input ASCII file. For this example, the data file has a defined record length of 40 bytes.

Figure 7-13
Format for Records in Input Sequential Files



Importing Records From an ASCII File

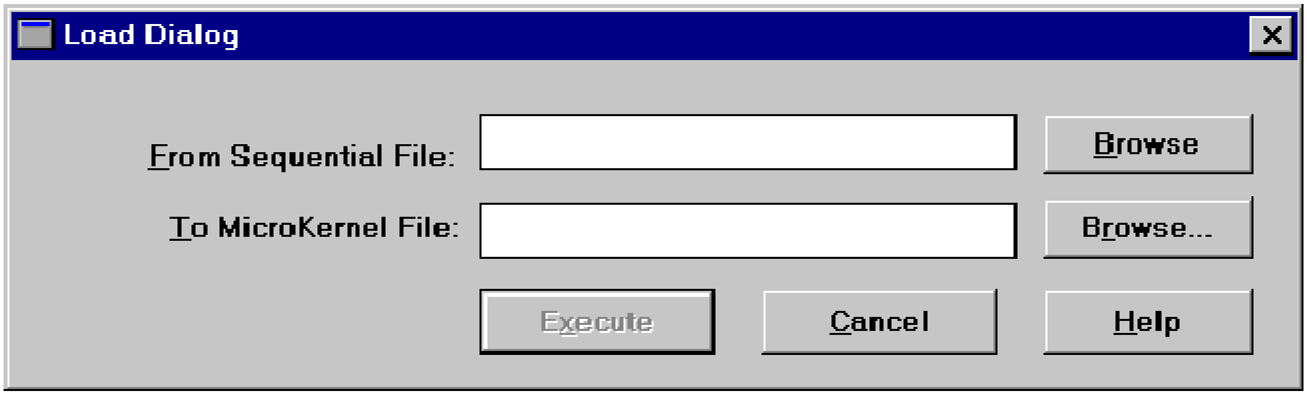
You can use the Maintenance utility to import records from an ASCII file to a standard data file. This operation does not perform any conversions on the data. You can create an import file using a text editor or the Maintenance utility (see "Exporting Records to an ASCII File").

To import ASCII data

1. Choose **Load** from the **Data** menu:

2. The **Load** dialog box (Figure 7-14) appears.

Figure 7-14
Load Dialog



The ASCII file you specify must adhere to the specifications explained in "Importing and Exporting ASCII File Format" . The record length of the standard data file you specify must be compatible with the records in the ASCII file.

3. Click **Execute** to import the records.

While importing data, the Maintenance utility shows the number of records being imported, the percentage of records imported, and a status message. You can continue working in the Maintenance utility (for example, you can open another **Load** dialog box).

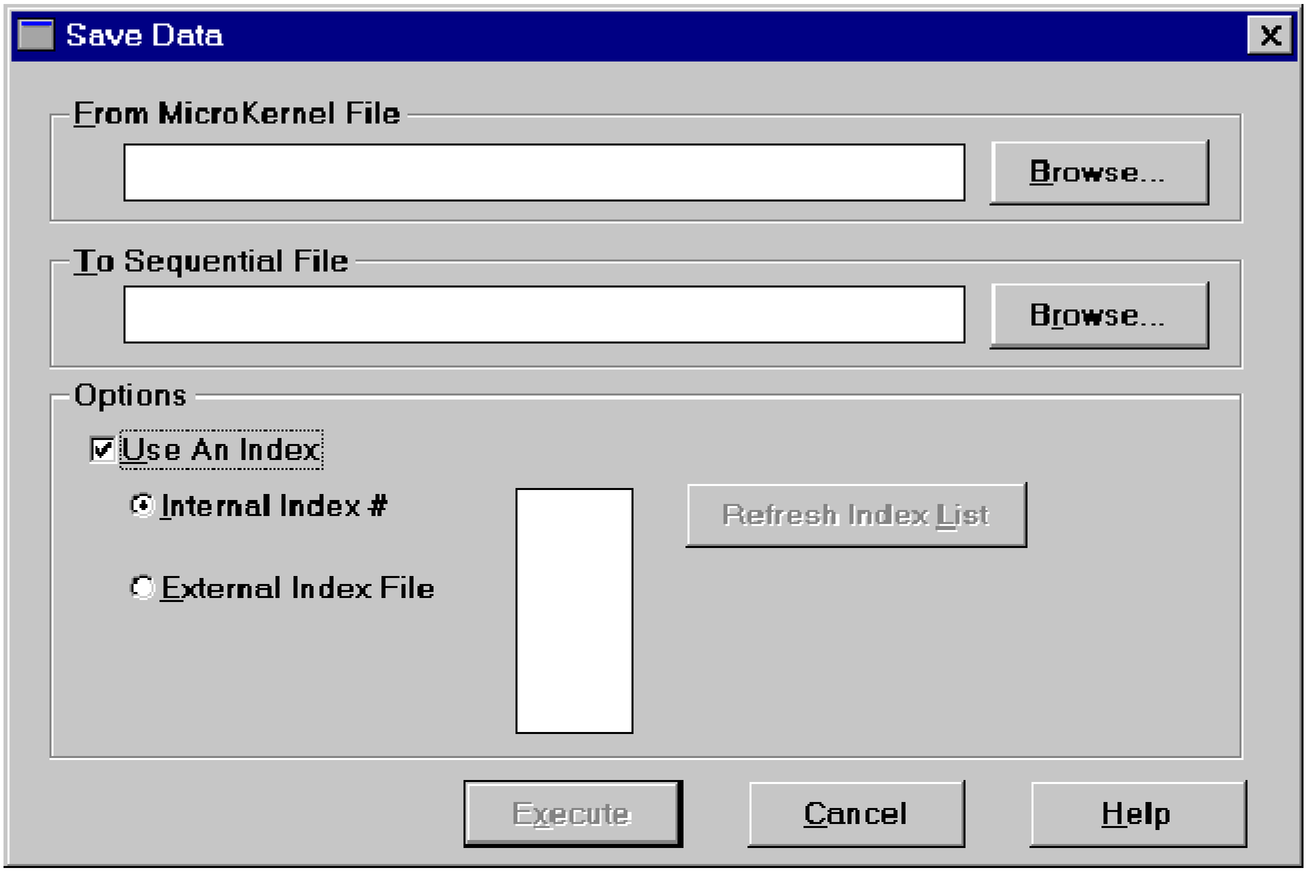
Exporting Records to an ASCII File

You can use the Maintenance utility to export records from a data file to an ASCII file.

To export ASCII records:

1. Choose **Save** from the **Data** menu. The **Save Data** dialog box (Figure 7-15) appears.

Figure 7-15
Save Data Dialog



☑ Use An Index

● Internal Index #

○ External Index File

2. In the **Save Data** dialog box, specify the following options.

From MicroKernel File

Specifies the name of the existing MicroKernel-compatible file you want to save.

To Sequential File

Specifies the name of the sequential file to create.

Use An Index

Uses a specified index when sorting the records for export. By default, the Maintenance utility does not use an index, meaning that records are exported according to their physical position in the data file.

Internal Index # :
Uses the specified key number. Click **Refresh Index List** to update the available indexes if you change file in the **From MicroKernel File** box.

External Index File :
Uses the specified external index. (To create an external index, refer to “Creating Indexes”.)

Direction

Forward: This is the default setting and indicates the utility recovers the file from the beginning.Backward: This option recovers data from the end of the file.Forward and Backward: This option reads the file forward until it fails. Then it starts at the end of the file and reads the file backward until it reaches the record that failed previously or encounters another failure.Backward and Forward: Indicates the utility reads the file backward until it fails. Then it starts at the beginning of the file and reads the file forward until it reaches the record that failed previously or encounters another failure.

3. Click **Execute** to export the data. The Maintenance utility creates the specified ASCII file using the format described in "Importing and Exporting ASCII File Format" . You can then edit the ASCII file and use the **Load** command to import the edited text to another standard data file, as described in "Importing Records From an ASCII File".

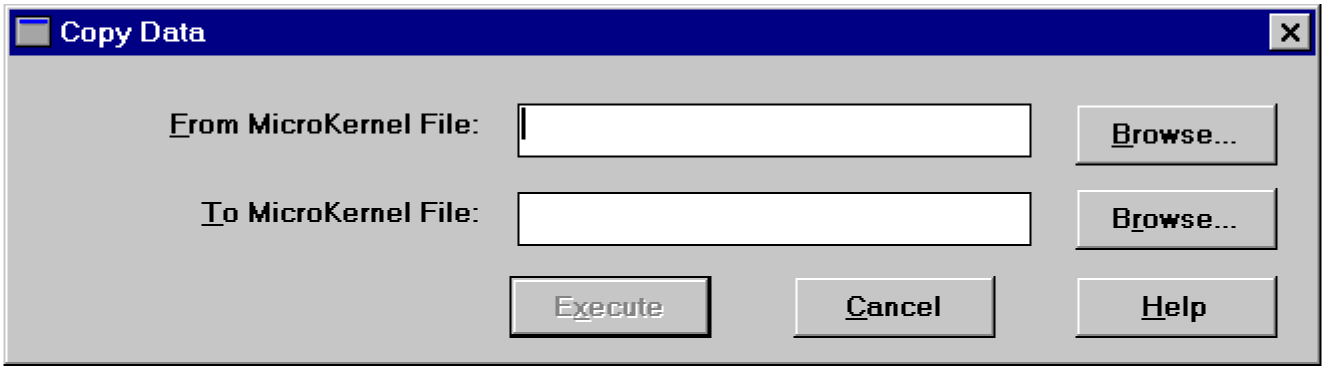
Copying Records Between Data Files

You can use the Maintenance utility to copy data from one standard data file to another.

To copy data:

1. Choose **Copy** from the **Data** menu. The Copy Data dialog box (Figure 7-16) appears.

Figure 7-16
Copy Data Dialog



2. Enter the name of the file you want to copy in the **From MicroKernel File** box and then specify the path where you want to copy the file in the **To MicroKernel File** box.

   The record lengths for both data files you specify must be the same.

Recovering Data After a System Failure

The Roll Forward feature enables you to recover changes made to a data file between the time of the last backup and a system failure. The MicroKernel stores the changes in an archival log file. If a system failure occurs, you can restore the backup copy of your data file and then use the Roll Forward feature, which applies all changes stored in the log to your backup copy.

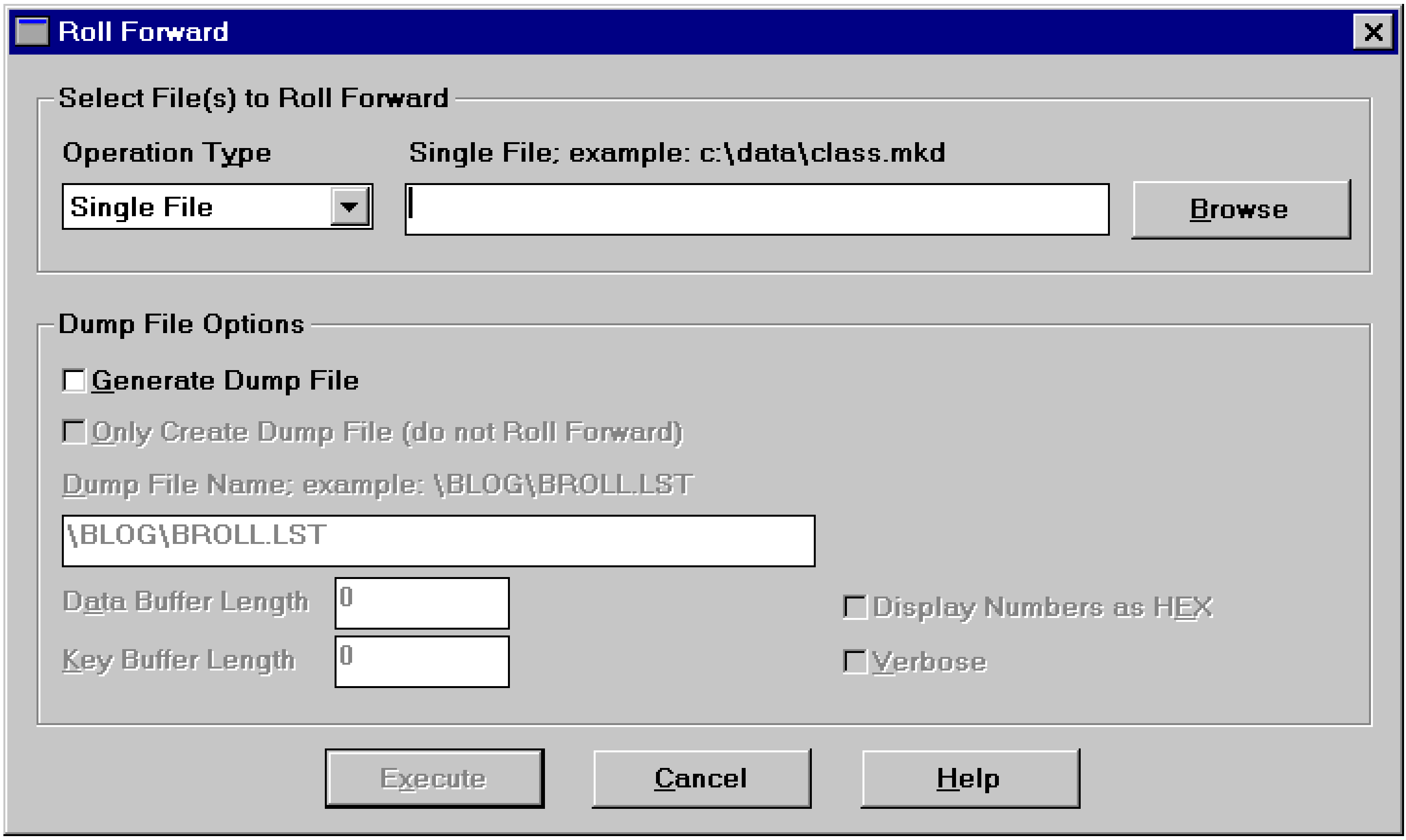
> **Note:** You cannot take advantage of the Roll Forward feature command unless you both enable the MicroKernel's "Archival Logging of Selected Files" option *and* back up your files *before* a system failure occurs. For information about backing up your files, refer to "Backing Up Your Files".

If a system failure occurs, restore your backup, start the MicroKernel, and immediately perform the Roll Forward feature. You must run the Roll Forward feature command *before* you access the files. Doing so guarantees that the data written to the data files is consistent up to the point of the system failure. In particular, you must perform the roll forward before you write to, lock, or get an exclusive handle on any of the files.

To perform a roll forward operation:

1. Select **Roll Forward** from the **Data** menu. The **Roll Forward** dialog (Figure 7-17) appears.

Figure 7-17
Roll Forward Dialog



Roll Forward

Select File(s) to Roll Forward

Operation Type: Single File

Single File; example: c:\data\class.mkd

Dump File Options

□ Generate Dump File

□ Only Create Dump File (do not Roll Forward)

Dump File Name; example: \BLOG\BROLL.LST

\BLOG\BROLL.LST

Data Buffer Length: 0

Key Buffer Length: 0

□ Display Numbers as HEX

□ Verbose

2. Select the specific operation type: single file, list of files, volume name, or drive letter. When you select either volume name or drive letter, you must insert a back slash (\) or forward slash (/) at the end (for example, \\server\vol1\ or D:\).

3. You can generate a log file, called a dump file, of all the Btrieve operations required to perform the roll forward tasks.

   By default, this file is not created. Select the **Generate Dump File** check box to generate a file. You can also specify the following options:

Only Create Dump File	Indicates that only the dump file is to be created, and the roll forward operation is not to be performed.
Dump File Name	Contains the name of the dump file, which must begin with a

slash and not contain a drive letter or server/volume name.

Data Buffer Length

Indicates the number of data buffer bytes to write to the dump file for each Btrieve operation.

Key Buffer Length

Indicates the number of key buffer bytes to write to the dump file for each Btrieve operation.

Display Numbers as HEX

If you select this option, the numbers in the dump file output are formatted as hexadecimal. If you do not select this checkbox, the numbers are displayed in ASCII format.

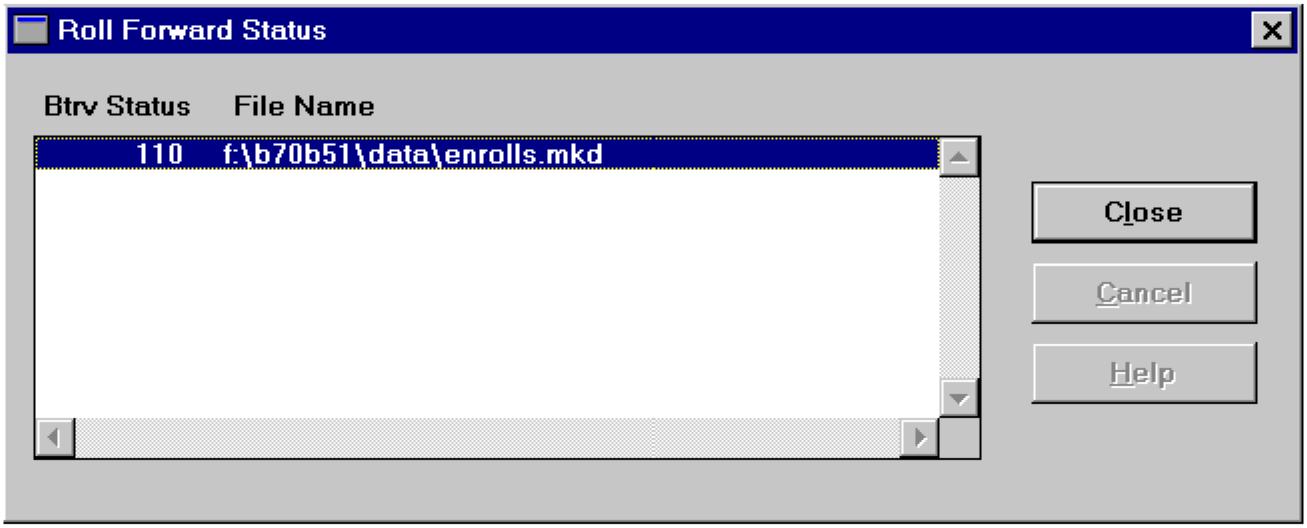
Verbose

Includes additional information like user name, network address, and time stamp in the dump file.

> **Note:** If the key buffer or the data buffer is not an input parameter for the particular Btrieve operation, nothing is written to the dump file.

4. Click **Execute** to generate the dump file and/or perform the roll forward operation. If the data is valid, the **Roll Forward Status** dialog (Figure 7-18) box appears.

Figure 7-18
Roll Forward Status Dialog



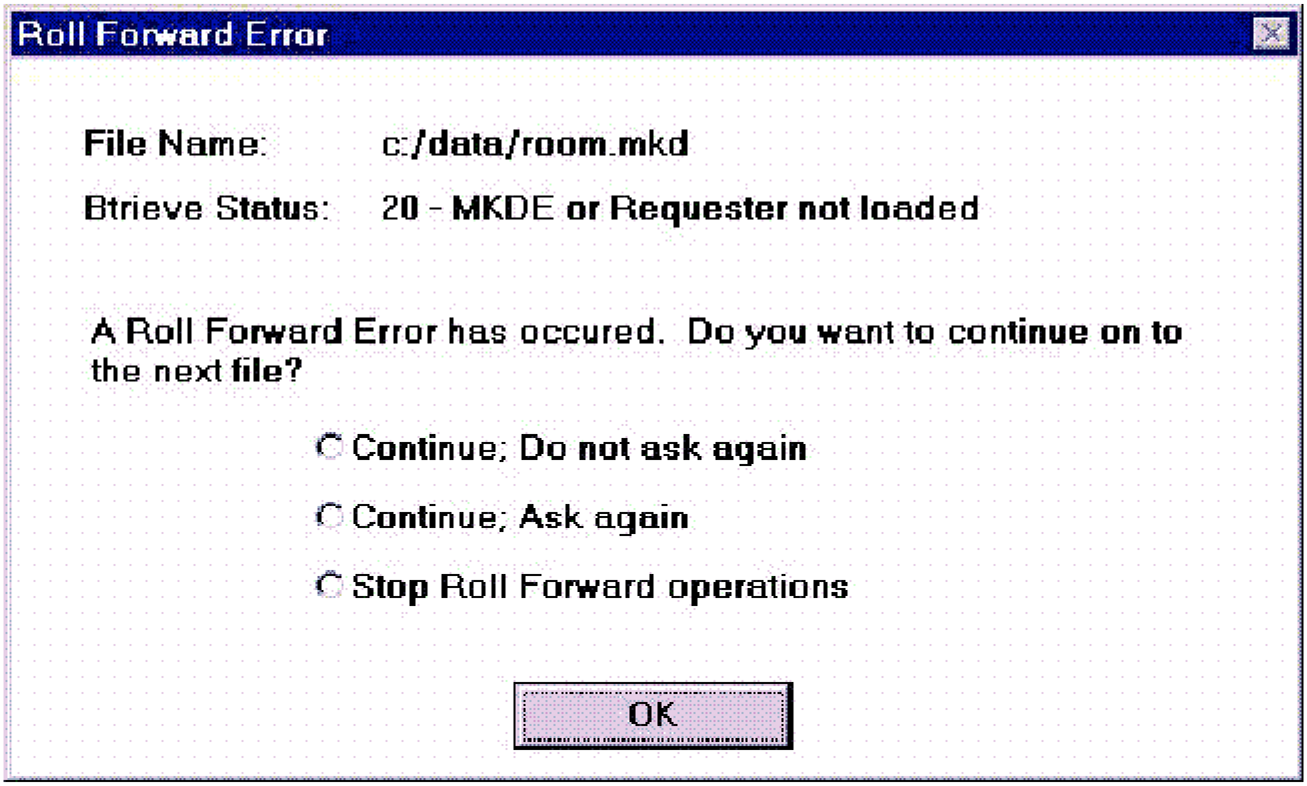
Roll Forward Status

Btrv Status	File Name
110	f:\b70b51\data\enrolls.mkd

As files are processed, they are added to the scrolling list box which displays the file name and the Pervasive.SQL status code returned from the roll forward operation.

If an error occurs during processing, the **Roll Forward Continue on Error** dialog box appears. This dialog allows you to continue without being prompted again, to continue and be prompted again, or to stop processing files.

Figure 7-19
File Format for Import ASCII Files



Roll Forward Error

File Name: c:/data/room.mkd

Btrieve Status: 20 - MKDE or Requester not loaded

A Roll Forward Error has occured. Do you want to continue on to the next file?

○ Continue; Do not ask again

○ Continue; Ask again

○ Stop Roll Forward operations

Btrieve Command-Line Maintenance Utility (BUTIL)

Use this utility if you prefer a command-line interface or if you want to start or stop continuous operation. The Btrieve Maintenance utility is also available in a command-line format that runs on the server (as an NLM on NetWare or from a DOS command prompt on Windows NT) or locally on DOS, Win32 and OS/2 clients. You can execute Maintenance utility commands from the command line or through a command file you create. Before you perform commands in the Btrieve Maintenance utility, also referred to as BUTIL, it is important you understand some concepts and elements addressed in the

The Btrieve Command-Line Maintenance utility performs the following common file and data manipulations:

- "Importing and Exporting Data"
- "Creating and Modifying Data Files"
- "Backing Up a Database"
- "Recovering Changes After a System Failure"
- "Viewing Data File Statistics"
- "Displaying Btrieve Interface Module Version"
- "Unloading the Btrieve Engine and Requester (DOS only)"

BUTIL Overview

This section provides information you need to know before using the command-line Maintenance utility commands. It discusses the following:

- "Commands"
- "Command Format"
- "BUTIL Concepts"

Commands

Table 7-6
Command-Line Maintenance Utility Commands

Command	Description
CLONE	Creates a new, empty data file using an existing file's specifications.
CLROWNER	Clears the owner name of a data file.
COPY	Copies the contents of one data file to another.
CREATE	Creates a data file.
DROP	Drops an index.
ENDBU	Ends continuous operation on data files defined for backup in Win32 and NLM versions only.
INDEX	Creates an external index file.
LOAD	Loads the contents of an unformatted file into a data file.
RECOVER	Reads data sequentially from a data file and writes the results to an

	unformatted file. (The DOS version does not support ROLLFWD.) Use this command if you have a damaged file.
ROLLFWD	Recovers changes made to a data file between the time of the last backup and a system failure.
SAVE	Reads data along a key path and writes the results to a sequential file.
SETOWNER	Assigns an owner name to a data file.
SINDEX	Creates an index.
STARTBU	Starts continuous operation on files defined for backup in Win32 and NLM versions only.
STAT	Reports statistics about file attributes and current sizes of data files.
(DOS only)	Unloads the Btrieve engine and requester.
VER	Displays the version of the MicroKernel Database Engine and Btrieve Interface Module that is loaded at the server.

Viewing Command Usage Syntax

To view a summary of each command usage, enter the following command at the file server:

```
BUTIL
```

The utility displays usage syntax for each command as illustrated in Figure 7-20.

Figure 7-20
Maintenance Utility Command Screen on NetWare

The command syntax is as follows:

```
BUTIL -CLONE <outputFile> <sourceFile> [/O<owner | *>] [/S]

BUTIL -CLROWNER <sourceFile> </O<owner | *>> [/S]

BUTIL @commandFile [commandOutputFile]

BUTIL -COPY <sourceFile> <outputFile> [/O<owner1 | *> [/O<owner2 | *>]] [/S]

BUTIL -CREATE <outputFile> <descriptionFile> [Y|N] [/S]

BUTIL -DROP <sourceFile> <keyNumber | SYSKEY> [/O<owner |*>] [/S]

BUTIL -ENDBU </A | sourceFile | @listFile> [/S]

BUTIL -INDEX <sourceFile> <indexFile> <descriptionFile> [/O<owner |*>] [/S]

BUTIL -LOAD <unformattedFile> <outputFile> [/O<owner |*>] [/S]
BUTIL -RECOVER <sourceFile> <unformattedFile> [/O<owner |*>] [/S]

BUTIL -ROLLFWD <sourceFile | volume | drive | @listFile> [</L[dumpFile]
| /W[dumpFile]> [/T<dataLength>] [/E<keyLength>] [/H] [/V]]
[/O<ownerList | owner | *>] [/A] [/S]

BUTIL -SAVE <sourceFile> <unformattedFile> [Y indexFile | N
<keyNumber | -1>] [/O<owner1 |*> [/O<owner2 |*>]] [/S]

BUTIL -SETOWNER <sourceFile> </O<owner |*>> <level> [/S]

BUTIL -SINDEX <sourceFile> <descriptionFile | SYSKEY> [keyNumber]
[/O<owner |*>] [/S]

BUTIL -STARTBU <sourceFile | @listFile> [/S]

BUTIL -STAT <sourceFile> [/O<owner |*>] [/S]

BUTIL -VER [/S]
```

Note: The /S option applies only to the NetWare version of the command-line utility. Also, on NetWare you always have to specify the full path of the file name such as sys:\demodata\tuition.mkd.

Command Format

The format for the Maintenance utility command line is as follows:

BUTIL [-*command* [*parameter* ...]] | @*commandFile*

–command A Maintenance utility command, such as COPY. You must precede the command with a dash (–), and you must enter a space before the dash. Table 7-6 lists the commands.

parameter Information that the command may require. Discussions of the individual commands provide details when applicable.

commandFile fully qualified file name of a command file.

BUTIL Concepts

Command Files

You can use a command file to do the following:

- Execute a command that is too long to fit on the command line.

- Execute a command that you use often (by entering the command once in the command file and then executing the command file as often as you want).

- Execute a command and write the output to a file, using the following command format:

```
BUTIL @commandFile [commandOutputFile]
```

For each command executed, the resulting output file shows the command followed by its results. All messages appear on the server console screen, as well.

- Execute multiple commands sequentially.

Command files contain the same information as that required on the command line.

Rules for Command Files

Observe the following rules when creating a Maintenance utility command file:

- You cannot split a single parameter across two lines.

- You must end each command with <end> or [end] .

Command File Example

The following is an example command file, COPYCRS.CMD. The file calls the BUTIL -CLONE command to create the NEWCRS.MKD file by cloning the COURSE.MKD file, and the -CREATE command to create the NEWFILE.DTA file by using the description provided in the NEWFILES.DES description file.

```
-clone newcrs.mkd course.mkd <end>
-create newfile.dta newfiles.des <end>
```

The following command uses the COPYPATS.CMD file and writes the output to the COPYPATS.OUT file:

```
butil @copypats.cmd copypats.out
```

Description Files

Description files are ASCII text files that contain descriptions of file and key specifications that the Maintenance utility can use to create data files and indexes. Some users employ description files as a vehicle for archiving information about the data files they have created. For more information about the description file format, see Appendix B, "Description Files."

Extended File Support

The size of the MicroKernel data file can be larger than the operating system file size limit. When you export data from an extended MicroKernel file to an unformatted file, the size of the unformatted file can exceed the MicroKernel file size limit because of the differences in the physical format.

The Interactive Maintenance utility detects that the unformatted file has exceeded the file size limit (2 GB) and starts creating extension files. This process is transparent. Extension files and the original unformatted file must reside on the same volume. The extension file uses a naming scheme similar to the MicroKernel Database Engine. The first extension file is the same base file name with '.~01' extension. The second extension file is '.~02,' and so on. These numbers are printed in hex. The maximum number of extension files is 255; the 255th extension file (the last extension file) has an extension of '.~ff'.

To SAVE or RECOVER huge files to unformatted files, see the respective command. Also, when you import data from an unformatted file, the utility detects if the file has extensions and loads the data from the extension file.

Owner Names

The MicroKernel allows you to restrict access to files by specifying an owner name. Because owner names are optional, the files you use with the utility may or may not require an owner name. If the file requires an owner name, you must specify it using the /O option. You can specify one of the following:

- Single owner name.
- List of up to eight owner names. Separate the owner names with commas.
- Asterisk (*). The utility prompts you for the owner name. With the ROLLFWD command, the utility prompts you for a list of owner names separated by commas.

Owner names are case-sensitive; Sandy and SANDY are not considered to be the same. If you enter owner names on the command line, the utility discards leading blanks. If you specify an asterisk, the utility does not discard leading blanks.

Redirecting Error Messages

Be sure that you specify a fully qualified file name (including a drive letter or UNC path) when redirecting error messages.

To redirect error messages to a file on a Windows NT server:

- Use the following command format.

```
BUTIL -command commandParameters > filePath
```

To redirect error messages to a file on a NetWare server:

- Use the following command format.

```
LOAD BUTIL -command commandParameters (CLIB_OPT)
/>filepath
```

ASCII File Format

See “Importing and Exporting ASCII File Format” in the Interactive Maintenance utility section.

Rules for Specifying File Names on Different Platforms

When you run BUTIL for Windows NT/95 or OS/2, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

Importing and Exporting Data

This section provides detailed information on importing and exporting data using the following BUTIL commands: COPY, LOAD, RECOVER, and SAVE.

Table 7-7
Commands to Import and Export Data

Command	Description
COPY	Copies the contents of one data file to another.
LOAD	Loads the contents of a sequential file into a data file.

RECOVER	Reads data sequentially from a data file and writes the results to a sequential file.
SAVE	Reads data along a key path and writes the results to a sequential file.

COPY

The COPY command copies the contents of one MicroKernel file to another. COPY retrieves each record in the input data file and inserts it into the output data file. The record size must be the same in both files. After copying the records, COPY displays the total number of records inserted into the new data file.

> **Note:** COPY performs in a single step the same function as a RECOVER command followed by a LOAD command.

Using the COPY command, you can create a data file that contains data from an old file, but has new key characteristics.

To copy a MicroKernel data file:

1. Use the CREATE command to create an empty data file with the desired key characteristics (key position, key length, or duplicate key values).

   or

   Use CLONE to create an empty data file using the characteristics of an existing file.

2. Use the COPY command to copy the contents of the existing data file into the newly created data file.

Format

BUTIL -COPY <_sourceFile_ > <_outputFile_ > [/O<_owner1_ |*>
[/O<_owner2_ |*>]] [/S]

sourceFile — The fully qualified name of the data file from which to transfer records. When you run BUTIL for Windows NT/95 or OS/2, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

outputFile — The fully qualified name of the data file into which to insert records. The output data file can contain data or be empty. When you run BUTIL for Windows NT/95 or OS/2, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

/Oowner1 — The owner name of the source data file, if required. If only the output data file requires an owner name, specify /O followed by a blank for _owner1_ (as illustrated in the example).

/Oowner2 — The owner name of the output data file, if required.

/S (NetWare only) — By default, the Maintenance utility stops at each full screen of output and waits for a keystroke before continuing. With the /S option, the utility continuously scrolls output on the screen. You cannot use /S on the command line if you specify a command file, but you can specify /S with a command inside a command file.

Example

The following command copies the records in COURSE.MKD to NEWCRS.MKD. The COURSE.MKD input file does not require an owner name, but the NEWCRS.MKD output file uses the owner name Pam.

```
butil -copy course.mkd newcrs.mkd /o /oPam
```

If you omit the first /O from this example, the utility assumes that the owner name Pam belongs to the input data file, not the output data file.

LOAD

The LOAD command inserts records from an input ASCII file into a file. The input ASCII file can be a single file or an extended file (the base file plus several extension files). LOAD performs no conversion on the data in the input ASCII file. After the utility transfers the records to the data file, it displays the total number of records loaded.

> **Note:** The LOAD command opens the output file in Accelerated mode; during a LOAD operation, the MicroKernel does not log the file. If you are using archival logging, back up your data files again after using the LOAD command.
>
> *Extended files:* If the utility finds the next extension file, it continues the load process. Do not delete any extension file created earlier by the SAVE and RECOVER commands. If the file has three extensions and the user deletes the second one, LOAD stops loading records after processing the first extension file. SAVE or RECOVER created three extension files and a fourth one exists from a previous SAVE or RECOVER, LOAD reads the records from the fourth extension and inserts them into the MicroKernel file. If a fourth file exists, then you need to delete it before starting the LOAD process.

Before running the LOAD command, you must create the input ASCII file and the data file. You can create the input ASCII file using a standard text editor or an application; the input ASCII file must have the required file format (as explained in "Importing and Exporting ASCII File Format"). You can create the data file using either the CREATE or the CLONE command.

Format

BUTIL -LOAD <*unformattedFile* > <*outputFile* > [/O<*owner* |*>]
[/S]

unformattedFile The fully qualified name of the ASCII file containing the records to load into a data file. For Windows NT, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

outputFile The fully qualified name of the data file into which to insert the records. When you run BUTIL for Windows NT/95 or OS/2, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

/Oowner The owner name for the data file, if required.

/S (NetWare only) By default, the Maintenance utility stops at each full screen of output and waits for a keystroke before continuing. With the /S option, the utility continuously scrolls output on the screen. You cannot use /S on the command line if you specify a command file, but you can specify /S with a command inside a command file.

Example

The following example loads sequential records from the COURSE.TXT file into the COURSE.MKD file. The owner name of the COURSE.MKD file is Sandy.

```
butil -load course.txt course.mkd /oSandy
```

RECOVER

The RECOVER command extracts data from a MicroKernel file and places it in an ASCII file that has the same format as the input ASCII file that the LOAD command uses. This is often useful for extracting some or all of the data from a damaged MicroKernel file. The RECOVER command may be able to retrieve many, if not all, of the file's records. You can then use the LOAD command to insert the recovered records into a new, undamaged MicroKernel file.

> **Note:** The Maintenance utility performs no conversion on the data in the records. Therefore, if you use a text editor to modify an output file containing binary data, be aware that some text editors may change the binary data, causing the results to be unpredictable.

Format

BUTIL -RECOVER <*sourceFile* > <*unformattedFile* > [/O<*owner* |*>] [/S] [/Q] [/J] [/I]

sourceFile — The fully qualified name of the data file from which to recover data. When you run BUTIL for Windows NT/95 or OS/2, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

unformattedFile — The fully qualified name of the ASCII file where the utility should store the recovered records.

/Oowner — The owner name for the data file, if required.

/S (NetWare only) — By default, the Maintenance utility stops at each full screen of output and waits for a keystroke before continuing. With the /S option, the utility continuously scrolls output on the screen. You cannot use /S on the command line if you specify a command file, but you can specify /S with a command inside a command file.

/Q — Indicates whether to replace an existing unformatted file. By default, the Maintenance utility overwrites the existing files. If you specify this option and a file with the same name exists, the utility returns an error message.The utility also checks whether the MicroKernel file to be recovered is extended. If the file is extended, the utility checks for files with the same name as the potential unformatted extension file. If one of those files exists, the utility returns an error message.

/J — Indicates BACKWARD reading of the file. If you specify this option, the utility recovers data from the MicroKernel file using STEP LAST and PREVIOUS operations. The default is forward reading, using STEP FIRST and NEXT operations.

/I

Indicates FORWARD reading of the file. Although the default is forward reading, you can use this option to indicate FORWARD and BACKWARD reading. This means that if you specify both /I and /J, respectively, the utility reads the file forward until it fails. Then it starts at the end of the file and reads backwards until it reaches the record that failed previously or encounters another failure.If you specify /J first, the utility reads backwards and then reads forward.

For each record in the source file, if the RECOVER command receives a variable page error (Status Code 54), it places all the data it can obtain from the current record in the unformatted file and continues the recovery process.

The utility produces the following messages:

- informs you about the name of the last extension file created
- checks if the next extension file exists, and if so, tells you to delete it
- if you move the extended unformatted files to a different location, you are prompted to move the base file and all of its extension files

Example

The following example extracts records from the COURSE.MKD file and writes them into the COURSE.TXT file.

```
butil -recover course.mkd course.txt
```

SAVE

The SAVE command retrieves records from a MicroKernel file using a specified index path and places them in an ASCII file that is compatible with the required format for the LOAD command. You can then edit the ASCII file and use the LOAD command to store the edited data in another data file. (See “Importing and Exporting ASCII File Format”for more information about the ASCII file format.)

SAVE generates a single record in the output ASCII file for each record in the input data file. Upon completion, SAVE displays the total number of records saved.

> **Note:** The Maintenance utility performs no conversion on the data in the records. Therefore, if you use a text editor to modify an output file containing binary data, be aware that some text editors may change the binary data, causing the results to be unpredictable.

Format

```
BUTIL -SAVE <sourceFile > <unformattedFile > [Y indexFile
| N <keyNumber | -1>] [/O<owner1 |*> [/O<owner2 |*>]] [/S] [/Q] [/J] [/I]
```

sourceFile

The fully qualified name of the data file containing the records to save. When you run BUTIL for Windows NT/95 or OS/2, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

unformattedFile

The fully qualified name of the ASCII file where you want the utility to store the records.

indexFile
The fully qualified name of an external index file by which to save records *if* you do not want to save records using the default of the lowest key number.

keyNumber
The key number (other than 0) by which to save records *if* you do not want to save records using the default of the lowest key number.

-1
The specification for saving the records in physical order using the Btrieve Step operations.

/Oowner1
The owner name for the source file, if required. If only the index file requires an owner name, specify /O followed by a blank for *owner1* .

/Oowner2
The owner name for the index file, if required.

/S (NetWare only)
By default, the Maintenance utility stops at each full screen of output and waits for a keystroke before continuing. With the /S option, the utility continuously scrolls output on the screen. You cannot use /S on the command line if you specify a command file, but you can specify /S with a command inside a command file.

/Q
Indicates whether to replace an existing unformatted file. By default, the Maintenance utility overwrites the existing files. If you specify this option and a file with the same name exists, the utility returns an error message.The utility also checks whether the MicroKernel file to be saved is extended. If the file is extended, the utility checks for files with the same name as the potential unformatted extension files. If one of those files exists, the utility returns an error message.

/J
Indicates BACKWARD reading of the file. If you specify this option, the utility recovers data from the MicroKernel file using GET LAST and PREVIOUS operations. The default is forward reading, using GET FIRST and NEXT operations.

/I
Indicates FORWARD reading of the file. Although the default is forward reading, you can use this option to indicate FORWARD and BACKWARD reading. This means that if you specify both /I and /J, respectively, the utility reads the file forward until it fails. Then it starts at the end of the file and reads backwards until it reaches the record that failed previously or encounters another failure.If you specify /J first, the utility reads backwards and then reads forward.

The utility produces the following messages:

- informs you about the name of the last extension file created
- checks if the next extension file exists, and if so, tells you to delete it
- if you move the extended unformatted files to a different location, you are prompted to move the base file and all of its extension files

Examples

The following two examples illustrate how to use the SAVE command to retrieve records from a data file.

This example uses a NEWCRS.IDX external index file to retrieve records from the COURSE.MKD file and store them in an unformatted text file called COURSE.TXT:

```
butil save course.mkd course.txt newcrs.idx
```

The following example retrieves records from the COURSE.MKD file using key number 3 and stores them in an unformatted text file called COURSE.TXT:

```
butil -save course.mkd course.txt n 3
```

Creating and Modifying Data Files

This section includes detailed information on creating and modifying data files using the following BUTIL commands: CLONE, CLROWNER, CREATE, DROP, INDEX, SETOWNER, and SINDEX. This section also includes information about removing unused space in a Btrieve data file, which is discussed in “Compacting Btrieve Data Files”

Table 7-8
Commands to Create and Modify Data Files

Command	Description
CLONE	Creates a new, empty data file using an existing file’s specifications.
CLROWNER	Clears the owner name of a data file.
CREATE	Creates a data file.
DROP	Drops an index.
INDEX	Creates an external index file.
SETOWNER	Assigns an owner name to a data file.
SINDEX	Creates an index.

CLONE

The CLONE command creates a new, empty file with the same file specifications as an existing file (including any supplemental indexes, but excluding the owner name). The new data file includes all the defined key characteristics (such as key position, key length, or duplicate key values) contained in the existing file.

The CLONE command ignores all MicroKernel configuration options that affect file statistics (such as “System Data”) *except* file version. The CLONE command creates a new file using the MicroKernel file version you specify with the Create File Version option.

Format

```
BUTIL -CLONE <outputFile > <sourceFile > [/O<owner | *>] [/S]
```

outputFile — The fully qualified file name to use for the new, empty data file. When you run BUTIL for Windows NT/95 or OS/2, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

sourceFile — The fully qualified file name of the existing data file to replicate.When you run BUTIL for Windows NT/95 or OS/2, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

/Oowner — The owner name, if any, for the source data file. The new data file does not have an owner name. See “Owner Names”for more information.

/S (NetWare only) — By default, the Maintenance utility stops at each full screen of output and waits for a keystroke before continuing. With the /S option, the utility continuously scrolls output on the screen. You cannot use /S on the command line if you specify a command file, but you can specify /S with a command inside a command file.

Remarks

Btrieve 6.0 and later allows a maximum of 23 key segments in a data file with a page size of 1,024 bytes. Therefore, the CLONE command sets the page size in the new data file to 2,048 bytes if the existing data file contains 24 key segments and has a page size of 1,024 bytes. This occurs if the existing data file has a format earlier than 6.0 and the MicroKernel was not loaded with the “Create File Version” option set to 5.*x* or 6.*x* .

If you are cloning a pre-7.*x* file, ensure that the MicroKernel is configured to create the file format version that you want the new file to be. For example, if you want to clone a 6.15 file in 7.*x* format, ensure that the MicroKernel File Format Version option is set to 7.*x* .

> **Note:** If your source file is in 7.x format and it does not contain system data, your output file will not contain system data, regardless of the MicroKernel configuration. To add system data to an existing file, refer to *Getting Started with Pervasive.SQL.*

If you are trying to recover from receiving Status Code 30 (The file specified is not a MicroKernel file) and you suspect that the header page of the source file might be damaged, try creating the new MicroKernel file using the CREATE command with a description file.

Example

The following command creates the NEWCRS.MKD file by cloning the COURSE.MKD file.

```
butil -clone newcrs.mkd course.mkd
```

CLROWNER

The CLROWNER command clears the owner name of a MicroKernel file.

Format

BUTIL -CLROWNER <*sourceFile* > </O<*owner* |*>> [/S]

sourceFile — The fully qualified file name of the data file. When you run BUTIL for Windows NT/95 or OS/2, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

/Oowner — The owner name to clear. See “Owner Names” for more information.

/S (NetWare only) — By default, the Maintenance utility stops at each full screen of output and waits for a keystroke before continuing. With the /S option, the utility continuously scrolls output on the screen. You cannot use /S on the command line if you specify a command file, but you can specify /S with a command inside a command file.

Example

The following command clears the owner name for the TUITION.MKD file. The owner name for the file is Sandy.

```
butil -clrowner tuition.mkd /oSandy
```

CREATE

The CREATE command generates an empty MicroKernel file using the characteristics you specify in a description file. Before you can use the CREATE command, you must create a description file to specify the new key characteristics. For more information, see Appendix B, "Description Files."

Format

BUTIL -CREATE <*outputFile* > <*descriptionFile* > [Y|N] [/S]

outputFile — The fully qualified file name of the MicroKernel file to create. If the file name is the name of an existing MicroKernel file, this command creates a new, empty file in place of the existing file. Any data that was stored in the existing file is lost and cannot be recovered. When you run BUTIL for Windows NT/95 or OS/2, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

descriptionFile — The fully qualified name of the description file containing the specifications for the new MicroKernel file.

Y|N — Indicates whether to replace an existing file. If you specify N but a MicroKernel file with the same name exists, the utility returns an error message. The default is Y.

/S (NetWare only) — By default, the Maintenance utility stops at each full screen of output and waits for a keystroke before continuing. With the /S option, the utility continuously scrolls output on the screen. You cannot use /S on the command line if you specify a command file, but you can specify /S with a command inside a command file.

Example

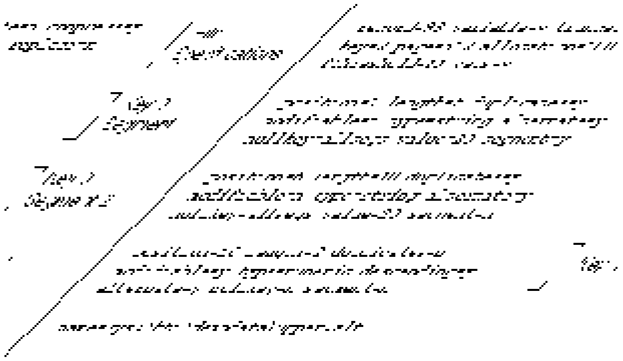
The following command creates a file named COURSE.MKD using the description provided in the CREATE.DES description file.

```
butil -create course.mkd create.des
```

Sample Description File for the CREATE Command

The sample description file shown in Figure 7-21creates a MicroKernel formatted file. The file is specified to have a page size of 512 bytes and 2 keys. The fixed-length portion of each record in the file is set to 98 bytes. The file specifies variable-length records with no blank truncation, data compression, and variable-tail allocation tables (VATs). The free space threshold is set to 20 percent. Allocation is set to 100 pages. The MicroKernel preallocates 100 pages, or 51,200 bytes, when it creates the file.

Figure 7-21
Sample Description File for the CREATE Command



Key 0 is a segmented key with two duplicatable, nonmodifiable string segments and a null value of 20 hexadecimal (space) specified for both segments. Key 0 uses the collating sequence UPPER.ALT.

Key 1 is a numeric, nonsegmented key that does not allow duplicates but permits modification. It is sorted in descending order.

DROP

The DROP command removes an index from a file and adjusts the key numbers of any remaining indexes, subtracting 1 from each subsequent key number. If you do not want to renumber the keys, you can add 128 to the key number you specify to be dropped. This renumbering feature is available only for 6.0 and later files.

Format

BUTIL -DROP <*sourceFile* > <*keyNumber* | SYSKEY> [/O<*owner* |*>] [/S]

sourceFile The fully qualified name of the file from which you are dropping the index. When you run BUTIL for Windows NT/95 or OS/2, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

keyNumber The number of the key to remove. To preserve the original key numbers, add a 128 bias to the key number you specify.

SYSKEY Instructs the utility to drop the system-defined log key (also called system data). Dropping the system-defined log key does not delete values from the records; the MicroKernel still assigns unique system-defined log key values to newly inserted records.

However, the MicroKernel cannot perform logging for a file from which the system-defined log key is dropped, if no user-defined unique keys exist. For this reason, you should use this option only if you suspect that the system-defined log key is corrupt and you intend to re-add it.The SINDEX command allows you to re-use the system-defined log key once you have dropped it.

/Oowner The owner name for the file, if required.

/S (NetWare only) By default, the Maintenance utility stops at each full screen of output and waits for a keystroke before continuing. With the /S option, the utility continuously scrolls output on the screen. You cannot use /S on the command line if you specify a command file, but you can specify
/S with a command inside a command file.

Examples

In both of the following examples, COURSE.MKD has three keys. The original keys in the file are numbered 0, 1, and 2.

In the first example, the BUTIL -DROP command drops key number 1 from the COURSE.MKD file and renumbers the remaining key numbers as 0 and 1.

```
butil -drop course.mkd 1
```

In the following example, the BUTIL –DROP command drops key number 1, but does not renumber the keys. The key numbers remain 0 and 2.

```
butil -drop course.mkd 129
```

INDEX

The INDEX command builds an external index file for an existing MicroKernel file, based on a field not previously specified as a key in the existing file. Before you can use the INDEX command, you must create a description file to specify the new key characteristics. For more information about description files, see Appendix B, “Description Files.”

The records in the new file consist of the following:

- The 4byte address of each record in the existing data file.
- The new key value on which to sort.

> **Note:** If the key length you specify in the description file is 10 bytes, the record length of the external index file is 14 bytes (10 plus the 4byte address).

Format

BUTIL -INDEX <*sourceFile* > <*indexFile* > <*descriptionFile* >
[/O<*owner* |*>] [/S]

sourceFile
The fully qualified name of the existing file for which to build an external index. When you run BUTIL for Windows NT/95 or OS/2, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

indexFile
The fully qualified name of the index file in which the MicroKernel should store the external index.

descriptionFile
The fully qualified name of the description file you have created containing the new key definition. The description file should contain a definition for each segment of the new key.

/Oowner
The owner name for the data file, if required.

/S
(NetWare only)
By default, the Maintenance utility stops at each full screen of output and waits for a keystroke before continuing. With the /S option, the utility continuously scrolls output on the screen. You cannot use /S on the command line if you specify a command file, but you can specify /S with a command inside a command file.

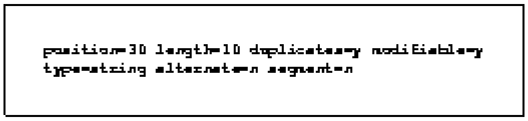
Remarks

The INDEX command creates the external index file and then displays the number of records that were indexed. To retrieve the data file’s records using the external index file, use the SAVE command.

Sample Description File for the INDEX Command

The description file shown in the following illustration defines a new key with one segment. The key begins at byte 30 of the record and is 10 bytes long. It enables duplicates, is modifiable, is a STRING type, and uses no alternate collating sequence.

Figure 7-22
Sample Description File for INDEX Command



```
position=30 length=10 duplicates=y modifiable=y
type=string alternate=n segment=n
```

Example

The following command creates an external index file called NEWCRS.IDX using a data file called COURSE.MKD. The COURSE.MKD file does not require an owner name. The description file containing the definition for the new key is called NEWCRS.DES.

```
butil -index course.mkd newcrs.idx newcrs.des
```

SETOWNER

The SETOWNER command sets an owner name for a MicroKernel file.

Format

BUTIL -SETOWNER <*sourceFile* > </O<*owner* |*>> < ***level*** > [/S]

sourceFile
The fully qualified name of the data file. When you run BUTIL for Windows NT/95 or OS/2, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

/Oowner
The owner name to be set

level
The type of access restriction for the data file. The possible values for this parameter are as follows

0: Requires an owner name for any access mode (no data encryption)

1: Permits read access without an owner name (no data encryption)

2: Requires an owner name for any access mode (with data encryption)

3: Permits read access without an owner name (with data encryption)

/S (NetWare only)
By default, the Maintenance utility stops at each full screen of output and waits for a keystroke before continuing. With the /S option, the utility continuously scrolls output on the screen. You cannot use /S on the command line if you specify a command file, but you can specify /S with a command inside a command file.

Example

The following example creates an owner for the COURSE.MKD file. The owner name is Sandy, and the restriction level is 1.

```
butil -setowner course.mkd /oSandy 1
```

SINDEX

The SINDEX command creates an additional index for an existing MicroKernel file. By default, the key number of the new index is one higher than the previous highest key number for the data file, or you can instruct the MicroKernel to use a specific key number. An exception is if a DROP command previously removed an index without renumbering the remaining keys, thus producing an unused key number; in this case, the new index receives the first unused number.

You can instruct the MicroKernel to use a specific key number for the new index with the key number option. The key number you specify must be a valid key number that is not yet used in the file. If you specify an invalid key number, you receive Status Code 6.

If you do not use the SYSKEY option with this command, you must create a description file that defines key specifications for the index before you can use the SINDEX command. For more information about description files, see Appendix B, “Description Files.”

Format

```
BUTIL -SINDEX <sourceFile > <descriptionFile | SYSKEY> [keyNumber ] [/O<owner   |*>] [/S]
```

sourceFile — The fully qualified name of the data file for which you are creating the index. When you run BUTIL for Windows NT/95 or OS/2, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

descriptionFile — The fully qualified name of the description file containing the description of the index to create.

SYSKEY — Instructs the utility to re-add the system key on a file in which the system key was dropped.

/Oowner — The owner name for the data file, if required.

/S (NetWare only) — By default, the Maintenance utility stops at each full screen of output and waits for a keystroke before continuing. With the /S option, the utility continuously scrolls output on the screen. You cannot use /S on the command line if you specify a command file, but you can specify /S with a command inside a command file.

Examples

The following example adds an index to the COURSE.MKD file. The name of the description file is NEWIDX.DES.

```
butil -sindex course.mkd newidx.des
```

The following example adds the system-defined key to the COURSE.MKD file. The system-defined key was dropped.

```
butil -sindex course.mkd syskey
```

Compacting Btrieve Data Files

You can use several commands in the BUTIL (CLONE, RECOVER, and LOAD, respectively) to remove unused space in a data file to decrease its size.

To compact a Btrieve data file:

1. Rename your data file and then use the CLONE option to create a blank data file using the original file name.
2. Use RECOVER to save the data from the clone file to an unformatted text file in sequential order.
3. Use LOAD to load the recovered data into the clone.

   Every record containing data will load into the newly created data file without blank records.

> **Note:** You can also perform this operation in the Btrieve Interactive Maintenance utility.

Backing Up a Database

This section provides detailed information on backing up a database using following BUTIL commands: STARTBU and ENDBU.

Table 7-9
Commands to Start and Stop Continuous Operation

Command	Description
STARTBU	Starts continuous operation on files defined for backup.
ENDBU	Ends continuous operation on data files defined for backup.

Continuous Operation

Continuous operation is a MicroKernel feature that enables you to back up files while they are in use by an application. During continuous operation, the MicroKernel creates a temporary data file (called a *delta* file) for each file in continuous operation to record any changes made to the data file while the backup is taking place. The temporary delta file may surpass the size of the original data file if users make extensive changes to the file during continuous operation.

Note: Temporary delta files have the same name as the data files but with a .^^^ extension. Therefore, do not create multiple data files with the same names but different extensions. For example, do not use a naming scheme such as INVOICE.HDR and INVOICE.DET for your data files.

When continuous operation ends, the MicroKernel updates the master data files with the changes stored in the delta files. The MicroKernel deletes the delta files when the master data files have been updated. You place files into continuous operation using STARTBU command. You end continuous operation on files using ENDBU command. The best time to place data files into continuous operation for backup is when the fewest users will be making modifications to the files.

Continuous operation mode does not significantly affect MicroKernel performance; however, using a server to back up files can affect performance.

If you are not using the MicroKernel's Archival Logging feature, perform the following steps. This procedure allows you to take advantage of the MicroKernel's continuous operation feature, which allows you to back up files while they are still in use by an application. However, if a system failure occurs, you cannot use the ROLLFWD command to recover changes since the last backup.

To protect against data loss using Continuous Operation:

1. Use the BUTIL -STARTBU command to put your files in continuous operation.
2. Back up your data files.
3. Use the BUTIL -ENDBU command to take your files out of continuous operation.

STARTBU

The STARTBU command places a file or set of files into continuous operation for backup purposes. You can perform this command on a Pervasive.SQL server engine with the Win32 and NLM versions of the utility, or a workstation engine on Windows 9*X* and NT platforms.

To back up files using continuous operation:

1. Issue the STARTBU command, followed by the data file or set of data files.
2. Run your backup program.
3. Issue the ENDBU command to stop continuous operation.

Format

BUTIL -STARTBU <*sourceFile* | @*listFile* > [/S]

sourceFile — The fully qualified name of the data file (including the drive specification for Windows NT and volume specification for NetWare) on which to begin continuous operation for backup.

listFile — The name of a text file containing the fully qualified names of files on which to begin continuous operation. Separate these file names with a carriage return/line feed. (Although the utility accepts a blank space separator as well, future versions of Pervasive.SQL may accept blank characters in file names. For compatibility with future versions of Pervasive.SQL, use the carriage return/line feed separator.)If the Maintenance utility cannot put all of the specified files in continuous operation, the utility does not put any of the files in continuous operation.

/S (NetWare only) — By default, the Maintenance utility stops at each full screen of output and waits for a keystroke before continuing. With the /S option, the utility continuously scrolls output on the screen. You cannot use /S on the command line if you specify a command file, but you can specify /S with a command inside a command file.

Note: This command begins continuous operation only on the files you specify. You cannot use wildcards with the STARTBU command.

Examples for Windows NT Server

The first example starts continuous operation on the COURSE.MKD file.

For Windows NT:

```
butil -startbu f:\demodata\course.mkd
```

The following example starts continuous operation on all files listed in the STARTLST.FIL file.

```
butil -startbu @startlst.fil
```

The STARTLST.FIL file might consist of the following entries:

```
f:\demodata\course.mkd
f:\demodata\tuition.mkd
f:\demodata\dept.mkd
```

Examples for NetWare Server

The first example starts continuous operation on the COURSE.MKD file.

```
butil -startbu sys:\demodata\course.mkd
```

The following example starts continuous operation on all files listed in the STARTLST.FIL file.

```
butil -startbu @sys:\test\startlst.fil
```

The STARTLST.FIL file might consist of the following entries:

```
sys:\demodata\course.mkd
sys:\demodata\tuition.mkd
sys:\demodata\dept.mkd
```

ENDBU

The ENDBU command ends continuous operation on a data file or set of data files previously defined for backup. Issue this command after using the STARTBU command to begin continuous operation and after performing your backup.

You can perform this command on a Pervasive.SQL server engine with the Win32 and NLM versions of the utility, or a workstation engine on Windows 9*X* and NT platforms.

Format

BUTIL -ENDBU </A |*sourceFile* | *@listFile* > [/S]

/A
If you specify /A , the utility stops continuous operation on all data files initialized by BUTIL –STARTBU and currently running in continuous operation mode.

sourceFile
The fully qualified name of the data file (including the drive specification for Windows NT and volume specification for NetWare) for which to end continuous operation.

@listFile
The name of a text file containing a list of data files for which to end continuous operation. The text file must contain the fully qualified file name for each data file, and you must separate these file names with a carriage return/line feed. (Although the utility accepts a blank space separator as well, future versions of Pervasive.SQL may accept blank characters in file names. For compatibility with future versions of Pervasive.SQL, use the carriage return/line feed separator.)Typically, this list of data files is the same as the list used with the STARTBU command.

/S (NetWare only)
By default, the Maintenance utility stops at each full screen of output and waits for a keystroke before continuing. With the /S option, the utility continuously scrolls output on the screen. You cannot use /S on the command line if you specify a command file, but you can specify /S with a command inside a command file.

Example for Windows NT Server

The following example ends continuous operation on the COURSE.MKD file.

```
butil -endbu f:\demodata\course.mkd
```

However, you can also just enter `butil -endbu course.mkd` instead of the full path if your current directory is f:\demodata.

Example for NetWare Server

The following example ends continuous operation on the COURSE.MKD file.

```
butil -endbu sys:\demodata\course.mkd
```

Recovering Changes After a System Failure

This section provides detailed information on recovering changes made to a data file between the time of the last backup and a system failure using the ROLLFWD command.

Archival Logging

You can configure the MicroKernel to perform archival logging, which can facilitate your file backup activities. If a system failure occurs, you can use the archival log files and the BUTIL -ROLLFWD command or Roll Forward feature in the interactive Maintenance utility to recover changes made to a file between the time of the last backup and the system failure. You turn archival logging on using the Archival Logging of Selected Files option in the Configuration utility.

You specify the files for which you want the MicroKernel to perform archival logging by adding entries to an archival log configuration file you create on the volume that contains the files. To set up the configuration file, follow these steps:

1. Create a \BLOG directory in a real root directory of the physical drive that contains data files you want to log. (That is, do not use a mapped root directory.) If your files are on multiple volumes, create a \BLOG directory on each volume.

2. In each \BLOG directory, create an empty BLOG.CFG file. You can use any text editor to create the BLOG.CFG file.

3. In each BLOG.CFG file, create entries for the data files on that drive for which you want to perform archival logging. The entries in the log file must be in 6.x format. Use the following format to create the entries:

```
\path1\dataFile[=\path2\logFile]
```

path1 The path to the data file to be logged. The path cannot include a drive letter.

dataFile The name of the data file to be logged.

path2 The path to the log file. Because the log file and the data file can be on different drives, the path can include a drive letter.

logFile The name of the log file.

A single entry cannot contain spaces and must fit completely on one line. (Each line can contain up to 256 characters.) If you have room, you can place multiple entries on the same line; separate each entry with at least one space.

If you do not provide a name for the log file, the MicroKernel assigns the original file name plus a .LOG extension to the log file when you first open it. For example, for the file B.BTR, the MicroKernel would assign the name B.LOG to the log file.

The following examples show three sample entries in the BLOG.CFG file on drive C. All three entries produce the same result: activity in the file C:\DATA\B.BTI is logged to the file C:\DATA\B.LOG.

```
\data\b.bti

\data\b.bti=\data\b.log

\data\b.bti=c:\data\b.log
```

The next example directs the engine to log activity in the file C:\DATA\B.BTI to the log file D:\DATA\B.LGF. This example shows that archival log files do not have to reside on the same drive as the data file and do not require a .LOG extension. (The .LOG extension is the default.)

```
\data\b.bti=d:\data\b.lgf
```

Backing Up Your Files

Backing up your files regularly is an important step in protecting your data.

If you are using the MicroKernel's "Archival Logging" feature, perform the following steps. This procedure allows you to take advantage of the ROLLFWD command or the Roll Forward feature in the interactive Maintenance utility if a system failure occurs.

To protect against data loss using Archival Logging:

1. Turn on the MicroKernel's Archival Logging of Selected Files option.

2. Create BLOG.CFG log configuration files for the volumes containing data files you want to log (as discussed in "Archival Logging").

3. Shut down the MicroKernel.

4. Back up your data files and delete existing log files. Deleting the existing log files keeps the archival log files from becoming too large. After each backup, delete the corresponding log files *before* you resume working with the data files. Synchronizing the backup data files and the corresponding log files is critical to recovering operations successfully.

5. Restart the MicroKernel.

If you are not using the MicroKernel's archival logging feature, perform the following steps. This procedure allows you to take advantage of the MicroKernel's Continuous Operation feature, which allows you to back up files while they are still in use by an application. However, if a system failure occurs, you cannot use BUTIL -ROLLFWD to recover changes since the last backup.

ROLLFWD

The ROLLFWD command recovers changes made to a data file between the time of the last backup and a system failure. The MicroKernel stores the changes in an archival log file. If a system failure occurs, you can restore the backup copy of your data file and then use the ROLLFWD command, which applies all changes stored in the log to your backup copy.

> **Note:** You cannot take advantage of the ROLLFWD command unless you both enable the MicroKernel's Archival Logging of Selected Files option (see "Archival Logging of Selected Files") *and* back up your files *before* a system failure occurs.

If a system failure occurs, restore your backup and immediately run the ROLLFWD command. You must run the ROLLFWD command before you access the files. Doing so guarantees that the data written to the data files is consistent up to the point of the system failure. In particular, you must perform the roll forward before you write to, lock, or get an exclusive handle on any of the files.

You can also use the ROLLFWD command to produce an output file of logged operations. The ROLLFWD command can produce the output file either before you roll changes forward or at the same time as the roll forward.

Format

```
BUTIL -ROLLFWD <sourceFile | volume | drive | @listFile >
[ </L[dumpFile ] | /W[dumpFile ]> [/T<dataLength> ]
[/E<keyLength> ]   [/H] [/V] [/O<ownerList >|<owner >|*]]
[/A] [/S]
```

sourceFile

The fully qualified name of the data file for which to roll forward changes. For Windows NT, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

volume

A volume for which to roll forward changes. End the volume name with a backslash (\) or forward slash (/), as in SYS:\ , //SERVER/SYS/ , or \\SERVER\SYS:\ .

drive

A drive letter for which to roll forward changes. End the volume name with a backslash (\) or forward slash (/), as in F:\ or F:/ .

listFile

The fully qualified name of a text file containing the paths of files, volumes, or drives for which to roll forward changes. Separate these paths with a carriage return/line feed. If the Maintenance utility encounters an error, the utility stops rolling forward the current file, but does not roll back the changes already made. If you specify the /A option, the utility continues rolling forward with the next file.

/L

Produces an output file, but does not roll forward.

/W

Rolls forward and produces an output file.

dumpFile

The file name of the output file to which the Maintenance utility writes a list of logged operations. The default is \BLOG\BROLL.LST. The file name cannot contain a drive letter or volume name and must start with a forward slash (/) or backslash (\). The Maintenance utility places the file on the same volume as the BLOG.CFG file.

/TdataLength

Specifies the length of the operation's data buffer to write to the output file. If you do not specify this option, the utility does not include data buffer contents in the output file.

/EkeyLength

Specifies the length of the operation's key buffer to write to the output file. If you do not specify this option, the utility does not include key buffer contents in the output file.

/H

Instructs the utility to show numbers in the output file in hexadecimal notation. If you do not specify this option, numbers in the output file are in ASCII format. This option affects the format of the Entry Count, Op Code, Key Number, and Data Length fields.

/V

Instructs the utility to include additional information (such as the user name, network address, and time stamp) in the output file.

/O

Specifies the owner name of the data file, if required. An owner name is required if you request an output file of logged operations and the backup copy of the data file has an owner name for read-only access. See "Owner Names" for more information.

/A

Specifies that if you are rolling back more than one file and the Maintenance utility encounters an error, the utility continues rolling forward with the next file.When you do not specify this option, the utility stops rolling forward if it encounters an error. (The utility does not roll back the changes already made.)**Note:** When you use the /A option, you might want to redirect output to

	a file, as described in “Redirecting Error Messages” and “Command Files”
/S (NetWare only)	By default, the Maintenance utility stops at each full screen of output and waits for a keystroke before continuing. With the /S option, the utility continuously scrolls output on the screen. You cannot use /S on the command line if you specify a command file, but you can specify /S with a command inside a command file.

> **Note:** If the key buffer or the data buffer is not an input parameter for the particular Btrieve operation, nothing is written to the dump file.

Examples

The following example recovers changes to the CLASS.MKD file.

```
butil -rollfwd sys:pvsw\demodata\class.mkd
```

This example recovers changes and outputs them to all files on the sys: volume with the following options:

- use default dump file
- dump 32 bytes of the data buffer
- dump 4 bytes of the key buffer
- dump in hex mode

```
butil -rollfwd sys:\ /W /H /T32 /E4
```

The following example does not perform roll forward but only outputs the changes to the files listed in files.txt with the following dump options:

- use sys:\temp\files.lst as the dump file
- use verbose mode
- data files have owner names: own123 and own321
- do not dump data or key buffer

```
butil -rollfwd @sys:\temp\files.txt /L\temp\files.lst /V /Oown123,own321
```

Viewing Data File Statistics

This section includes information about generating a report that contains a data file’s characteristics and statistics using STAT.

STAT

The STAT command generates a report that contains defined characteristics of a data file and statistics about the file’s contents. Using the STAT command is a good way to determine if a file can be logged by the MicroKernel’s transaction durability feature. The STAT command reports indexes the same whether they were created by the Create Supplemental Index operation (in Btrieve 6.0 and later) or the Create operation.

Format

BUTIL -STAT <*sourceFile* > [/O<*owner* |*>] [/S]

sourceFile The fully qualified name of the data file for which to report statistics. For Windows NT, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

/Oowner The owner name for the data file, if required.

/S
(NetWare only) By default, the Maintenance utility stops at each full screen of output and waits for a keystroke before continuing. With the /S option, the utility continuously scrolls output on the screen. You cannot use /S on the command line if you specify a command file, but you can specify /S with a command inside a command file.

Example

The following example reports file statistics for the PATIENTS.DTA file. The data file does not have an owner name.

```
butil -stat patients.dta
```

However for NetWare, you must specify the full path such as:

```
butil -stat sys:\demodata\patients.dta .
```

The following example shows the resulting report:

```
                        File Statistics for PATIENTS.DTA



 File Version = 7.00

 Page Size = 2048

 Page Preallocation = No

 Key Only = No

 Extended = No



 Total Number of Records = 16

 Record Length = 104

 Data Compression = No

 Variable Records = No
```

```
Available Linked Duplicate Keys = 0

Balanced Key = No

Log Key = 1

 System Data = No

Total Number of Keys = 3

Total Number of Segments = 4


Key Segment Position Length Type Flags Null Values* Unique ACS Values


0 1 21 20 String MD -- 16 0

0 2 7 12 String MD -- 16 0

1 1 1 6 String M -- 16 0

2 1 83 10 String MD -- 7 0

Alternate Collating Sequence(ACS) List:

 0 UPPER


Legend:

 < = Descending Order

 D = Duplicates Allowed

 I = Case Insensitive

 M = Modifiable

 R = Repeat Duplicate
```

A = Any Segment (Manual)

L = All Segments (Null)

* = The values in this column are hexadecimal.

?? = Unknown

-- = Not Specified

This example shows that the file called PATIENTS.DTA is a 7.0 file. (The version number indicates the earliest Btrieve version that can read the file format.) The file has a page size of 2,048 bytes and has no preallocated pages. This is not a key-only file, nor is it an extended file.

Sixteen records have been inserted into the file. The file was defined with a record length of 104 bytes, does not use data compression, and does not allow variable-length records.

There are no linked duplicate keys available in the file. The file does not use balanced indexing. The MicroKernel performs logging using Key 1, and the file contains no system-defined data. The file has three keys comprised of four key segments.

Note: Indexes created with SINDEX are designated with the letter R by default *unless* you specified the Reserved Duplicate Pointer element.

The STAT report also provides information about specific keys. For example, the report shows that Key 0 allows duplicates, is modifiable, and consists of two segments:

- The first segment starts in position 21, is 20 characters long, allows duplicates, is modifiable, and will be sorted as a STRING type. The dashes indicate that a null value was not defined. The Unique Values column indicates that 16 unique values were inserted for this segment. This segment uses the UPPER.ALT alternate collating sequence file.

- The second segment starts in position 7, is 12 characters long, allows duplicates, is modifiable, and will be sorted as a STRING type. Sixteen unique values were inserted for this segment. This segment uses the UPPER.ALT alternate collating sequence file.

Key 1 is the key the MicroKernel uses in logging this file. Key 1 consists of one segment. It starts in position 1, is six characters long, does not allow duplicates, is modifiable, and will be sorted as a STRING type. Sixteen unique values were inserted for this key. This key uses the UPPER.ALT alternate collating sequence file.

Key 2 consists of one segment. It starts in position 83, is 10 characters long, allows duplicates, is modifiable, and will be sorted as a STRING type. Seven unique key values were inserted for this key. This key uses the UPPER.ALT alternate collating sequence file.

Displaying Btrieve Interface Module Version

This section includes detailed information about displaying the version of the Btrieve Interface module using the VER command.

VER

The VER command returns the version number of both the MicroKernel and the Btrieve Access Module.

Format

```
BUTIL -VER [/S]
```

/S (NetWare only) By default, the Maintenance utility stops at each full screen of output and waits for a keystroke before continuing. With the /S option, the utility continuously scrolls output on the screen. You cannot use /S on the command line if you specify a command file, but you can specify /S with a command inside a command file.

Remarks

When you run the VER command, the utility displays messages similar to the following (for Windows NT):

```
The Btrieve Requester version is 7.00.
The Btrieve Version is 7.00.
```

Unloading the Btrieve Engine and Requester (DOS only)

STOP

Use the STOP command to unload the Btrieve engine and, if applicable, the requester.

Format

```
butil -stop
```

Manipulating Scalable SQL Data Files with the Maintenance Utility

The Scalable SQL Maintenance utility performs the following common file and data manipulations:

- "Scalable SQL Maintenance Utility Overview"
- "Importing and Exporting Data"
- "Backing Up a Database"
- "Recovering Changes After a System Failure"
- "Displaying SQL Interface Module Version"

Scalable SQL Maintenance Utility Overview

The Scalable SQL Maintenance utility is a command-line utility that runs on the server (as an NLM on NetWare or from a DOS command prompt on Windows NT) and workstation engines (Windows 95/98/NT). You can execute Maintenance utility commands from the command line or through a command file you create. Before you perform commands in the Scalable SQL Maintenance utility, it is important you understand some concepts and elements addressed in this section. It discusses the following:

- "Commands"
- "Command Format"
- "Command Options"
- "Scalable SQL Maintenance Utility Concepts"
- "Command Files"

Commands

Table 8-1
Scalable SQL Maintenance Utility Commands

Command	Description	Platform
BLOAD	Inserts records in bulk from a sequential file into a SQL data file.	NT, NetWare
COPY	Copies selected data from one data file to another data file using SQL statements to specify input and output.	NetWare only
ENDBU	Ends continuous operation on database names defined for backup.	NT, NetWare
LOAD	Loads and updates data from a sequential file into a data file using SQL statements to specify input.	NetWare only
ROLLFWD	Recovers changes made to data files within a database between the time of the last backup and a system failure.	NT, NetWare
SAVE	Saves selected data from a data file to a sequential file using SQL statements to specify export.	NetWare only
STARTBU	Starts continuous operation on database names defined for backup.	NT, NetWare
VER	Displays the version of the Scalable SQL Interface Module that is loaded at the server, including enhanced version information.	NetWare only

Note: Windows NT Users: Use the SQLScope utility to perform import and export operations.

Viewing Command Usage Syntax

To view a summary of each command usage, enter the following command at the file server:

```
sqlutil
```

The utility responds with usage syntax for each command.

NetWare

```
sqlutil -bload <TableName> <InputFile> <{UNF | SDF | ASC}>
 [/o:<ownerlist> | <owner> | <*> ] [options]

sqlutil @CommandFileName

sqlutil -copy <SQL InputFile> <SQL OutputFile>
 [/o:<ownerlist> | <owner> | <*> ] [options]

sqlutil -endbu <DatabaseName | @ListFile> [/s]

sqlutil -load <SQLFile> <InputFile> <{UNF | SDF | ASC}>
 [/o:<ownerlist> | <owner> | <*> ] [options]

sqlutil –rollfwd < DatabaseName | @ListFile   > [ </l:[dumpFile]
| /w:[dumpFile]> [/t:<dataLength> ] [/e:<keyLength> ] [/h] [/v]
 [/o:<ownerlist > | <owner > | <*> ]] [/a] [/s]

sqlutil -save <SQL file> <OutputFile> <{UNF | SDF | ASC}> [/o:<ownerlist>    | <owner> | <*> ] [options]

sqlutil -startbu <DatabaseName | @ListFile> [/s]

sqlutil -ver [/s]
```

Windows NT

```
sqlutil -bload <TableName> <InputFile> <{UNF | SDF | ASC}>
 [/o:<owner> | <*> ] [options]

sqlutil @CommandFileName

sqlutil -endbu <DatabaseName | @ListFile> [/s]

sqlutil –rollfwd < DatabaseName | @ListFile   > [ </l:[dumpFile] | /w:[dumpFile]> [/t:<dataLength> ]
 [/e:<keyLength> ] [/h] [/v][/o:<ownerlist > | <owner > | <*> ]] [/a] [/s]

sqlutil -startbu <DatabaseName | @ListFile> [/s]
```

Command Format

The format for the Scalable SQL Maintenance utility command line is as follows:

```
SQLUTIL [ < –command [ parameter ... ] | @commandFile >
[ option ... ] ]

option ::= < /B:specialBlankCharacter
| /C:
| /D:< dictionaryPath | databaseName >
| /F:dataFilePath
| /O:< ownerList | ownerName | *>
| /P:< userPassword | * >
| /R:numberOfRecords
| /S:
| /U:< username | * >
| /X:
>
```

–command A Scalable SQL Maintenance utility command, such as COPY. You must precede the command with a dash (–), and you must enter a space before the dash. Table 8-1 on page 8-3 lists the commands.

parameter Information that the command may require.

command_file path of a command file.

For a description of command options, refer to the following list.

Command Options

You can specify options in uppercase or lowercase. The options are as follows:

/B: Valid only in v3.01 compatibility mode. This option specifies the blank replacement character you want the utility to use. For example, to use a tilde (~) as the blank replacement character, specify the following:/B:~

/C: Tells the Scalable SQL Maintenance utility to run in v3.01 compatibility mode. By default, the Maintenance utility runs in v4.0 compatibility mode.This option is not valid on the input of the command file name but is a valid option for a command within a command file.

/D: Specifies either the database name or the directory in which the dictionary files reside. For example, to specify the university named database, enter /D:@BTU. To use the dictionary location, specify a path, such as /D:G:\SSQL\DEMODATA or /D:SYS:\SSQL\DEMODATA. The NLM version of this utility requires the /D: option. If you specify a named database, you do not need to use the /F: option to specify data file locations.

/F: Specifies the directory in which the data files reside, if necessary. Use this parameter only if you explicitly specify the dictionary location using the /D: option and the data file location is not fully defined in the dictionary. For example, if the data file location is specified in the dictionary only as FACULTY.MKD and the file is in the SSQL directory on drive G, specify /F:G:\SSQL. If you do not specify the /F: option and you do not specify a database name with the /D: option, the utility attempts to open the data file at the location stored in the dictionary.

/O: Specifies the owner name of the data file, if required. You can also specify a list of owner names, up to eight, separated by commas. See Owner Names for more information.

/P: Specifies the password, if necessary, of the user you named with the /U: option. Enter the password exactly as it is stored. If you want the utility to prompt you for the password, specify an asterisk (*) instead of a password.

/R: (For use with the BLOAD command only.) Specifies the number of records to load at one time. For example, to load 50 records at a time, specify /R:50. The default number of records is 100. In general, performance increases when you load larger numbers of records at one time. However, the block of records you load at one time cannot use more than 64 KB of memory.

/S By default, the Scalable SQL Maintenance utility stops at each full screen of output and waits for a keystroke before continuing. With the /S option, the utility continuously scrolls output on the screen. You cannot use /S on the command line if you specify a command file, but you can specify /S with a command inside a command file.

/U: Specifies a username, if necessary, for the dictionary you want to access. Specify a valid username for this option, such as /U:Charles. If you want the utility to prompt you for the username, specify an asterisk (*) instead of a username.

/X: Enables you to print only the total records processed (on –COPY, –LOAD, and –SAVE commands). When output from the Scalable SQL Maintenance utility is redirected, the utility also redirects the verifications from individual operations (such as inserts).

Scalable SQL Maintenance Utility Concepts

The following sections describe concepts you should understand before using the Scalable SQL Maintenance utility commands.

File Names

For commands that require a file path, you specify the volume name and directory path. The syntax is as follows:

NetWare Form	Windows NT Form
volume :\directoryPath \file name	*drive*:\directoryPath\file name

Owner Names

The MicroKernel allows you to restrict access to files by specifying an owner name. Because owner names are optional, the files you use with the utility may or may not require an owner name. If the file requires an owner name, you must specify it using the /O option. You can specify one of the following:

- Single owner name.
- List of up to eight owner names. Separate the owner names with commas. Except in the ROLLFWD command, if an owner name begins with a nonalphabetic character, enclose the owner name in single quotes.
- Asterisk (*)

Owner names are case-sensitive; Sandy and SANDY are not considered to be the same. If you enter owner names on the command line, the utility discards leading blanks. If you specify an asterisk, the utility prompts you for the owner name and does not discard leading blanks.

Redirecting Error Messages

To redirect error messages to a file, use one of the following command formats.

For NetWare, the command is as follows:

sqlutil -*command commandParameters* (CLIB_OPT)/> *filePath*

For Windows NT, the command is as follows:

```
sqlutil -commandcommandParameters > filePath
```

Be sure that you specify the full path when redirecting error messages.

Command Files

You can use a command file to do the following:

- Execute a command that is too long to fit on the command line.
- Execute a command that you use often (by entering the command once in the command file and then executing the command file as often as you want).
- Execute a command and write the output to a file, using the following command format:

```
SQLUTIL @commandFile [commandOutputFile ]
```

For each command executed, the resulting output file shows the command followed by its results. All messages appear on the screen, as well.

Command files contain the same information as that required on the command line.

Rules for Command Files

Observe the following rules when creating a Scalable SQL Maintenance utility command file:

- You must limit each line to 130 characters.
- You must limit the command file size to no more than 1,000 bytes.
- You cannot split a single parameter across two lines.
- Only one command is valid per command file; you cannot use multiple commands in a single command file.

Command File Example

The following is an example command file, LOADSTDT.CMD. The file calls the SQLUTIL –LOAD command to load data from a sequential file (STUDENTS.ASC) into a data file specified by the SQL statements contained in the STUDENTS.SQL file.

```
-load:\ssql\demodata\students.sql:\ssql\demodata\students.asc
/d:\ssql\demodata
/f:\ssql\demodata
```

The following command uses the LOADSTDT.CMD file to run the utility as an NLM on a file server. At the file server, you must specify the full path name of the command file, including the volume.

```
sqlutil @sys:\ssql\demodata\loadstdt.cmd
```

Importing and Exporting Data

This section provides detailed information on importing and exporting data using the following Scalable SQL Maintenance utility commands: BLOAD, COPY, LOAD, and SAVE.

Command	Description	Platform
BLOAD	Inserts records in bulk from a sequential file into a SQL data file.	NT, NetWare
COPY	Copies selected data from one data file to another data file using SQL statements to specify input and output.	NetWare only
LOAD	Loads and updates data from a sequential file into a data file using SQL statements to specify input.	NetWare only
SAVE	Saves selected data from a data file to a sequential file using SQL statements to specify export.	NetWare only

Note: Windows NT Users: Use the SQLScope utility to import and export data, which is discussed in "Importing and Exporting Data" .

Import and Export File Formats

The Scalable SQL Maintenance utility supports three different file formats for importing or exporting data. These formats are shown in Table 8-2.

Table 8-2
External Data Formats

Option	**Description**
UNF	Unformatted format. The utility does not convert the data to ASCII. Numeric columns remain in binary form and are not converted to printable ASCII characters. Each record is preceded by its length in ASCII and is followed by a comma delimiter. A carriage return/line feed terminates each record.
SDF	Standard Data Format. The data is represented as standard ASCII characters, the columns are separated by commas, and quotation marks enclose all columns. A carriage return/line feed terminates each record. This is a common format for exchanging data between applications or databases.
ASC	ASCII format. Each record in the input file must be preceded by the length of the record followed by a delimiter (either a comma or a space). The data follows the delimiter and should be the exact length given. Each column in the row must be the length of the display size of the column. Terminate each record with a carriage return/line feed.

Rules for Importing and Exporting Data

The following rules apply to importing and exporting data:

- When importing data, the Scalable SQL Maintenance utility does not accept keywords in SQL statements in place of substitution variables. You can import the data using substitution variables and then use SQLScope to perform an update on the columns using SQL statements that contain keywords.

- You can import or export variable-length columns (data types NOTE and LVAR) in unformatted (UNF) format only.

- The Scalable SQL Maintenance utility exports SDF data with double quotes (“) around all columns; this is required to handle data that contains a comma in the mask, when importing or exporting SDF data.

For more information about importing data, refer to the description of the INSERT statement in the *SQL Language Reference* .

BLOAD

The BLOAD command, which is available on NT and NetWare, inserts records in bulk from a sequential file into a SQL data file. (This operation drops indexes and re-adds them.) This provides a convenient way to transfer large amounts of data from a sequential file created by another program into a data file. The sequential file must contain input data for all the columns defined in the table; if you do not have input data for all columns, use the LOAD command.

You can import some data types using UNF format only. For more information, refer to “Rules for Importing and Exporting Data”“Rules for Importing and Exporting Data” on page 8-10.

When using the BLOAD command, data files are opened in Accelerated mode. This makes the BLOAD command faster than the LOAD command, but it also means that your transaction may not be durable if a failure occurs.

Format

SQLUTIL -BLOAD *table sequentialInputFile* < UNF | SDF | ASC >
[*options* ...]

table	The dictionary name of an existing Scalable SQL table into which to insert records.
sequentialInputFile	The sequential input file containing the records to load into the data file. This file must contain all the columns defined in the dictionary for the specified table.
< UNF | SDF | ASC >	The format of the data in the sequential input file.
options	Any utility options.

Example

The following command loads 10 records at a time from the ASCII file NEWSTDNT.ASC and inserts them into the Person table of the BTU database.

```
sqlutil -bload person newstdnt.asc asc /d:@btu /r:10
```

COPY

The COPY command, which is available only on NetWare, copies the contents of one data file to another. You can use the COPY command to change a column from one data type to a different, compatible data type. You can convert any data type to one of the string data types.

Using the COPY command, you can create a data file that contains data from an old file, but has new characteristics.

To create a data file with information from a historical file:

1. Create an empty data file with the desired dictionary definition using SQL statements in SQLScope.

2. Use the COPY command to copy the contents of the existing data file into the newly created data file.

You cannot copy data that contains variable-length data types (LVAR and NOTE). You must either export the data in UNF format and then import it in UNF format, or remove the column from the view in the SQL statement and then use the COPY command.

Format

SQLUTIL –COPY *SQLInputFile SQLOutputFile* [*options* ...]

SQLInputFile An ASCII text file that contains a valid SQL SELECT statement. This statement specifies the names of the columns to select, the SQL data file or files from which to select data, and any other join or restrict conditions to place on the SELECT statement.

SQLOutputFile An ASCII text file that contains a valid SQL INSERT or UPDATE statement with substitution variables. The substitution variable values you specify must correspond with the column names you specified in the SELECT statement.

option Any utility options.

Example

The following command copies data in the BTU named database using the previous input and output SQL files.

```
sqlutil -copy getnames.sql putnames.sql /d:@btu
```

This example copies the ID and Last Name of all records in the Students table to another, existing table called Names. It uses two files containing SQL statements. The input file, GETNAMES.SQL, contains the following text:

```
select id, last_namestudents
```

The output file, PUTNAMES.SQL, contains the following text:

```
insert into names (id, last_name)(@v1, @v2)
```

The SELECT statement in the input file supplies the values for the substitution variables V1 and V2 that the output file uses. For more information about using substitution variables, refer to the *SQL Language Reference* .

LOAD

The LOAD command inserts rows from an input sequential file into a SQL data file. This provides a convenient way to transfer data from a sequential file created by another program into a SQL file. The LOAD command is available on NetWare only.

> **Note:** Windows NT Users: Use the SQLScope utility to import and export data, which is discussed in "Importing and Exporting Data" .

This command performs the following actions:

1. Reads a SQL statement with substitution variables from an ASCII text file and passes it to Scalable SQL, which compiles it.

2. Reads data from an input file and passes it to Scalable SQL, which substitutes the values and executes the SQL statement.

As Scalable SQL loads data from the input file, the utility displays the total number of records loaded. The utility also displays a message when it has successfully loaded all records from the input file.

If you are loading data into NOTE or LVAR columns, your input file must be UNF format. If you are loading UNF data, you must use a SELECT statement.

For more information, refer to the *SQL Language Reference* .

The LOAD command opens files in Normal mode, so that your transaction is durable.

Format

SQLUTIL –LOAD *SQLFile inputFile* < UNF | SDF | ASC > [*option* ...]

SQLFile	A text file containing a valid INSERT, UPDATE, or DELETE statement with substitution variables. The SQL statement must specify the data file, the applicable column names, and the substitution variables. For the UNF format, you must use a SELECT statement, and you cannot specify substitution variables. For more information about SQL syntax, refer to the *SQL Language Reference* .
inputFile	The sequential file containing the data you want to load into the SQL data file.
<UNF|SDF| ASC>	The format of the data in the input file. For more information about sequential file format, refer to “ASCII File Format”.For more information about SDF data format, refer to “SDF File Format”
option	Any utility options, as described in “Command Options”

Example

The following command inserts a row into the Billing table in the BTU named database:

```
sqlutil –load amtowed.sql amtowed.sdf sdf /d:@btu
```

The SQL file, AMTOWED.SQL, contains the following statement:

```
insert into billing (student_id, amount_owed)(@v1, @v2);
```

The input file, AMTOWED.SDF, contains values for the substitution variables V1 and V2, as follows:

```
“116221385”,”$1500.00”
```

For more information about using substitution variables, refer to the *SQL* Language Reference.

SAVE

The SAVE command retrieves data from a database file and stores it in a sequential file. You can only perform this command on NetWare; however, Windows NT customers can use the SQLScope utility to import and export data as described in “Importing and Exporting Data” . You can use this command to extract data from one or more database files using a valid SQL SELECT statement. You can then edit the data and use the LOAD command to insert it into another database file.

SAVE generates a single row for each set of data the SELECT statement retrieves and exports the data in its current mask or the default mask, if one does not exist. You can specify the format of the output file with a command line parameter. As Scalable SQL inserts or saves the data into the output file, it displays the total number of rows saved.

> **Note:** The Scalable SQL Maintenance utility performs no conversion on the data in the rows. Therefore, if you use a text editor to modify an output file containing binary data, be aware that some text editors may change the binary data, causing the results to be unpredictable.

You can save variable-length data types using UNF format only. For more information, refer to “Rules for Importing and Exporting Data”.

Format

SQLUTIL –SAVE *SQLFile outputFile* < UNF | SDF | ASC > [*option* ...]

SQLFile The ASCII text file that contains a SQL SELECT statement. The SELECT statement is any valid SQL SELECT statement that does not contain substitution variables.

outputFile The file to which to SAVE the data.

< UNF | SDF | ASC > The format in which to store the data. For more information about sequential file format, refer to“ASCII File Format”For more information about SDF data format, refer to “SDF File Format”

option Any utility options, as described in “Command Options”

Example

The following example extracts data from the BTU named database and saves the extracted data to an SDF data file.

```
sqlutil -save amtowed.sql amtowed.sdf sdf /d:@btu
```

The SQL file, AMTOWED.SQL, uses the CAST function to export the data in the default column format, as follows:

```
select cast( student_id as type of student_id),
cast( amount_owed as type of amount_owed)billing
```

The resulting output file, AMTOWED.SDF, would contain the following values:

```
“116221385”,”$1500.00”
```

ASCII File Format

When you use the LOAD or SAVE command, records in an ASCII file have the following format. You can use an ASCII text editor to create import files, as long as they adhere to these specifications. Most text editors do not support editing binary data.

- The first column is a left-adjusted integer (in ASCII) that specifies the length of the record. (When calculating this value, ignore the carriage return/line feed that terminates each line.) The value in this first column matches the record length specified in the data file.
    - For files with fixed-length records, the length you specify should equal the record length of the data file.
    - For files with variable-length records, the length you specify must be at least as long as the minimum fixed length of the data file.
- A separator (a comma or a blank) follows the length column.

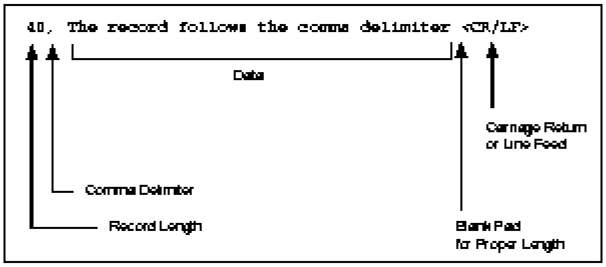
- The record data follows the separator. The length of the data is the exact number of bytes specified by the length column. If you are creating an import ASCII (sequential) file using a text editor, pad each record with blank spaces as necessary to fill the record to the appropriate length.

- A carriage return/line feed terminates each line. The Scalable SQL Maintenance utility does not insert the carriage return/line feed into the data file.

- The last line in the file is the end-of-file character (Ctrl+Z or 1A hexadecimal). Most text editors automatically insert this character at the end of a file.

Figure 8-1 shows the correct format for records in the input sequential file. For this example, the data file has a defined record length of 40 bytes.

Figure 8-1
Format for Records in Input Sequential Files



```
40, The record follows the comma delimiter <CR/LF>
```

SDF File Format

When you use the LOAD or SAVE command, records in an SDF (Standard Data Format) file have the following format. You can use an ASCII text editor to create import files, as long as they adhere to these specifications. Most text editors do not support editing binary data.

- All columns must be enclosed in double quotes (“ ”).

- A comma must separate all columns.

- A carriage return/line feed terminates each record.

The following example shows the correct format for records in the input sequential file:

“ART305”,”Sculpture”,”3” <CR/LF>

“ART406”,”Modern Art”,”3” <CR/LF>

“ART 407”,”Baroque Art”,”3” <CR/LF>

Backing Up a Database

This section provides detailed information on backing up databases using STARTBU and ENDBU commands.

Command	Description	Platform
STARTBU	Starts continuous operation on database names defined for backup.	NT, NetWare
ENDBU	Ends continuous operation on database names defined for backup.	NT, NetWare

STARTBU

The STARTBU command specifies a named database on which to begin continuous operation for backup purposes. To back up files using continuous operation, first issue the STARTBU command, followed by the database name or set of database names. Next, run your backup program. Then, issue the ENDBU command to stop continuous operation.

This command works locally on Pervasive.SQL server and workstation products.

Format

SQLUTIL –STARTBU < *databaseName* | *@listFile* > [/S]

databaseName — The database name on which to begin continuous operation for backup. This name must match a database name previously defined using the Scalable SQL Setup utility.

@listFile — The name of a text file containing a list of the database names on which to begin continuous operation. Separate these names with a carriage return/line feed.If the Scalable SQL Maintenance utility cannot put all of the files that make up the database name in continuous operation, the utility does not put any of the files in continuous operation.

Note: This command begins continuous operation only on the files within a database name you specify. For more information about continuous operation, see "Continuous Operation".

Examples

The following example starts continuous operation on the university database.

```
sqlutil -startbu btu
```

The following example starts continuous operation on all databases listed in the DBBACKUP.TXT file.

```
sqlutil -startbu @dbbackup.txt
```

ENDBU

The ENDBU command (available on both Windows NT and NetWare) ends continuous operation on a named database previously defined for backup. Execute this command after you have issued the STARTBU command and your backup utility has finished running. For more information about the STARTBU command, see page 8-18.

This command works locally on Pervasive.SQL server and workstation products.

Format

SQLUTIL –ENDBU < *databaseName* | *@listFile* > [/S]

databaseName The named database for which to end continuous operation.

@listFile The name of a text file containing the list of named databases for which to end continuous operation. The text file must contain the database names separated with a carriage return/line feed.

Typically, this list of data files is the same as the list used with the STARTBU command.

Example

The following example ends continuous operation on the university database.

```
sqlutil –endbu btu
```

Recovering Changes After a System Failure

This section provides detailed information on recovering data after a system failure has occurred using ROLLFWD.

Command	Description	Platform
ROLLFWD	Recovers changes made to data files within a database between the time of the last backup and a system failure.	NT, NetWare

ROLLFWD

The ROLLFWD command (available on both Windows NT and NetWare) recovers changes made to a data file between the time of the last backup and a system failure. The MicroKernel stores the changes in an archival log. If a system failure occurs, you can restore the backup copy of your data file and then use the ROLLFWD command, which applies all changes stored in the log to your backup copy.

Note: You cannot take advantage of the ROLLFWD command unless you both enable the MicroKernel's Archival Logging of Seleced Files option (see page 3-41) *and* back up your files *before* a system failure occurs. For information about backing up your files, refer to "Backing Up Your Files".

If a system failure occurs, restore your backup and immediately run the ROLLFWD command. You must run the ROLLFWD command before you access the files. Doing so guarantees that the data written to the data files is consistent up to the point of the system failure. In particular, you must perform the roll forward before you write to, lock, or get an exclusive handle on any of the files.

You can also use the ROLLFWD command to produce an output file of logged operations. The ROLLFWD command can produce the output file either before you roll changes forward or at the same time as the roll forward.

Format

```
SQLUTIL –ROLLFWD < databaseName | @listFile   > [ </L:[dumpFile ] | /W:[dumpFile ]>
[/T:<dataLength> ] [/E:<keyLength> ] [/H] [/V]
[/O:<ownerList > | <owner > | <*> ]] [/A] [/S]
```

databaseName	The name of the database for which to roll forward changes.
listFile	The path of a text file containing a list of database names for which to roll forward changes. Separate these database names with a carriage return/line feed.
/L:	Produces an output file, but does not roll forward.
/W:	Rolls forward and produces an output file.
dumpFile	The path of the output file to which the Scalable SQL Maintenance utility writes a list of logged operations. The default is /BLOG/BROLL.LST. The path cannot contain a drive letter or volume name and must start with a forward slash (\) or backslash (/). The Scalable SQL Maintenance utility places the file on the same

volume as the BLOG.CFG file.

/T:*dataLength* — Specifies the length of the operation’s data buffer to write to the output file. If you do not specify this option, the utility does not include data buffer contents in the output file.

/E:*keyLength* — Specifies the length of the operation’s key buffer to write to the output file. If you do not specify this option, the utility does not include key buffer contents in the output file.

Note: If the key buffer or the data buffer is not an input parameter for the particular Btrieve operation, nothing is written to the dump file.

/H — Instructs the utility to show numbers in the output file in hexadecimal notation. If you do not specify this option, numbers in the output file are in ASCII format. This option affects the format of the Entry Count, Op Code, Key Number, and Data Length fields.

/V — Instructs the utility to include additional information (such as the username, network address, and time stamp) in the output file.

/O: — Specifies the owner name of the data file, if required. An owner name is required if you request an output file of logged operations and the backup copy of the data file has an owner name for read-only access. See Owner Names for more information about owner names.#**Note:** With the ROLLFWD command, do not enclose owner names in single quotes, even if they begin with a nonalphabetic character.

/A — Specifies that if you are rolling back more than one database and the Scalable SQL Maintenance utility encounters an error, the utility continues rolling forward with the next database.When you do not specify this option, the utility stops rolling forward if it encounters any error other than Status Codes 12, 111, and 113. (The utility does not roll back the changes already made.)#**Note:** When you use the /A option, you may want to redirect output to a file, as described in “Redirecting Error Messages” and “Command Files”.

/S — By default, the Scalable SQL Maintenance utility stops at each full screen of output and waits for a keystroke before continuing. With the /S option, the utility continuously scrolls output on the screen. You cannot use /S on the command line if you specify a command file, but you can specify /S with a command inside a command file.

Examples

The following example rolls forward changes to the BTUDB database:

```
sqlutil -rollfwd btudb
```

The next example rolls forward changes to the BTUDB database and produces an output file of logged operations. The output file uses the default location (/BLOG/BROLL.LST) and includes the first 32 bytes of the data buffer and the first 4 bytes of the key buffer for each operation. The output file uses hexadecimal format.

```
sqlutil -rollfwd btudb /w /t:32 /e:4 /h
```

The next example does not roll forward any changes; it simply produces an output file of logged operations for the BTUDB database. The output file location is \TEMP\BTUDB.LST and includes a username, network address, and time stamp for each operation. The data files to be rolled forward have the owner names *Sandy* and *Tom* .

```
sqlutil -rollfwd btudb /l:\temp\btudb.lst /v /o:Sandy,Tom
```

Displaying SQL Interface Module Version

This section provides detailed information on displaying the version of the SQL Interface Module using VER.

Command	Description	Platform
VER	Displays the version of the Scalable SQL Interface Module that is loaded at the server, including enhanced version information.	NetWare only

VER

The VER command (on NetWare only) returns the version number of the Scalable SQL engine loaded at the server.

Format

```
SQLUTIL –VER [ /S ]
```

Remarks

When you run the VER command, the utility displays messages similar to the following:

```
Scalable SQL Version 4.0.0 is loaded.
```

The utility returns the version number, revision number, and patch level of the Scalable SQL engine that is loaded on the file server where the database name exists.

Executing SQL Statements with SQLScope

SQLScope allows you to execute SQL Statements interactively. This Win16 utility runs on Windows, Windows 95, and Windows NT operating systems. This chapter discusses the following functions you can perform with SQLScope:

- "SQLScope Overview"
- "Logging in to a Database"
- "Creating and Running SQL Statements"
- "Managing Referential Integrity"
- "Reviewing File-Level Information"
- "Importing and Exporting Data"
- "Recovering Damaged Data Files"

You can also set default login values and environmental settings, which is discussed in "Customizing SQLScope" later in this chapter.

SQLScope Overview

To start SQLScope:

- Click **Start** , highlight **Programs** , select **Pervasive SQL 7** , and choose **SQLScope (Win16)** . The **Login to Database** dialog appears (see Figure 9-1).

  If you want to use another method to start SQLScope, refer to your Windows, Windows 95, or Windows NT documentation for information about starting applications.

> **Note:** While SQLScope loads, it displays a dialog that contains information about the utility, including its version number. You can also display this information by choosing the **About** command from the **Help** menu.

Getting Help

To access the context-sensitive help system at any time while running SQLScope, do one of the following:

- Press F1–Depending on your cursor position, this key produces help information about the highlighted menu command, the current dialog, or SQLScope in general.
- Press Shift+F1–This key combination produces the help cursor, a question mark with an arrow. Use the help cursor to select the item for which you want help.
- Choose a command from the **Help** menu–The commands in the **Help** menu provide the following:
  - Information about how to use the Windows help system
  - An index of SQLScope help topics
  - Help for the active SQLScope window
  - A list of shortcut keys
  - Version and copyright information about SQLScope and version information for Scalable SQL and the MicroKernel.

Using the Shortcut Keys

When using SQLScope, you may find the shortcut keys useful for quickly performing certain tasks. Table 9-1 shows the shortcut key assignments.

Table 9-1
SQLScope Shortcut Keys

Press This Key	For This Action
Alt+F4	Exit SQLScope
F1	Get help for highlighted menu command or dialog box
Shift+F1	Enter help mode

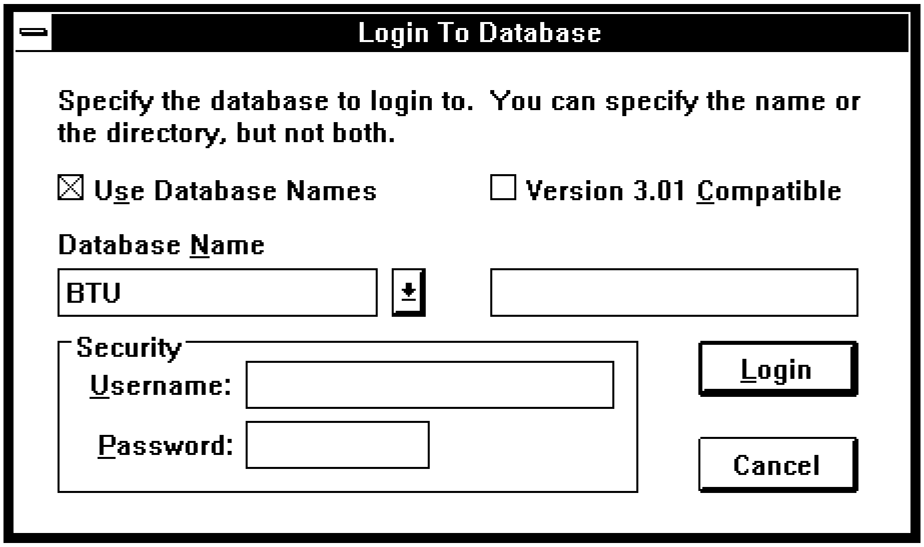
F5	Update database information in Tables or Columns list box
Ctrl+Z	Undo
Ctrl+X	Cut
Ctrl+C	Copy
Ctrl+V	Paste
Ctrl+S	Activate SQL text window
Ctrl+D	Activate database list
Ctrl+T	Activate table list
Ctrl+M	Activate column list
Ctrl+P	Activate template list
Ctrl+R	Activate results window
Ctrl+Shift+F	Move to first statement
Ctrl+Shift+P	Move to previous statement
Ctrl+Shift+N	Move to next statement
Ctrl +Shift+L	Move to last statement
Ctrl +F	Execute first statement
Ctrl+U	Execute current statement
Ctrl+ <keypad +>	Execute next statement
Ctrl+ <keypad ->	Execute previous statement
Ctrl+A	Execute all statements
Ctrl+O	Stop executing statements

Logging in to a Database

To log in to a database:

1. Start SQLScope (see page 9-2), which opens the **Login to Database** dialog as shown in Figure 9-1.

Figure 9-1
Login to Database Dialog



Specify the database to login to. You can specify the name or the directory, but not both.

☒ Use Database Names　□ Version 3.01 Compatible

Database Name: BTU

2. Specify either a database name or a directory that contains data dictionary files, but not both. You must log in to a database before you can perform most tasks in SQLScope.

   - If you log in by specifying a database name, select the **Use Database Names** check box, and then either select a name from the **Database Name** list, or enter the name of a database.

     The Database Name list contains all available database names on the local workstation and all available database names on the network if Scalable SQL is running on a file server. If you select a remote database, your engine usage configuration determines whether requests are processed locally or remotely.

   - If you log in by specifying a dictionary file location, clear the **Use Database Name** s check box, and then specify the location of the database's data dictionary files in the **Database Directory** text box. The location must be a valid, full path to a Scalable SQL data dictionary. You can use drive letters in paths to directories on servers.

3. If necessary, specify the compatibility mode.

   - If you log into a database that was created with Scalable SQL 4.0, clear the **Version 3.01 Compatible** check box.

     The compatibility mode check box allows you to log into Scalable SQL 3.01 databases and operate in 3.01 compatibility mode. If you log into a v4.0 database and you select this check box, you cannot use 4.0 syntax and data types in the database.

   - If you log into a database that was created with Scalable SQL 3.01 and you want the database to be compatible with applications written for 3.01, select the **Version 3.01 Compatible** check box.

If you log into a 3.01 database but do not select this check box, you must use v4.0 SQL syntax; if you then use v4.0 features, you could make the database incompatible with 3.01 statements and applications.

4. If necessary, specify your user name in the **Username** text box.

   If security is not enabled on the database you specified, the user name is not required.

5. If necessary, specify your password in the **Password** text box.

   If no password is defined for your user name or security is not enabled on the database, the password is not required. For security purposes, SQLScope does not display the password you specify; instead, it displays an asterisk (*) for each character.

6. Click **Login** or press Enter.

   If the login is successful, SQLScope Main window appears (see Figure 9-3). If the login fails, refer to the *Status Codes and* Messages manual for an explanation and suggested remedy for the error message or status code SQLScope displays.

Managing Multiple Logins

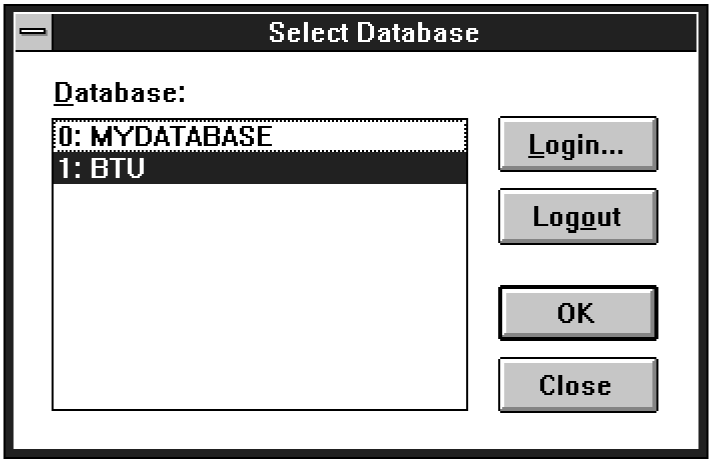
You can be logged in to more than one database at a time.

To log in to additional databases:

1. Either choose **Login** or from the **Database** menu, or choose from the **Database** menu and click **Login** in the **Select Database** dialog. Then follow the steps discussed in the previous section, "Logging in to a Database".

   Both the Database list in the SQLScope main window (shown in Figure 9-3) and the **Select Database** dialog (Figure 9-2) specify all current databases.

Figure 9-2
Select Database Dialog



0: MYDATABASE
1: BTU

2. To change the current database, select the desired database from either the Database list or the **Select Database** dialog.

Logging Out of Databases

To log out of a database:

1. Ensure that the database is shown in the Database list in the SQLScope main window (Figure 9-3 on page 9-8).

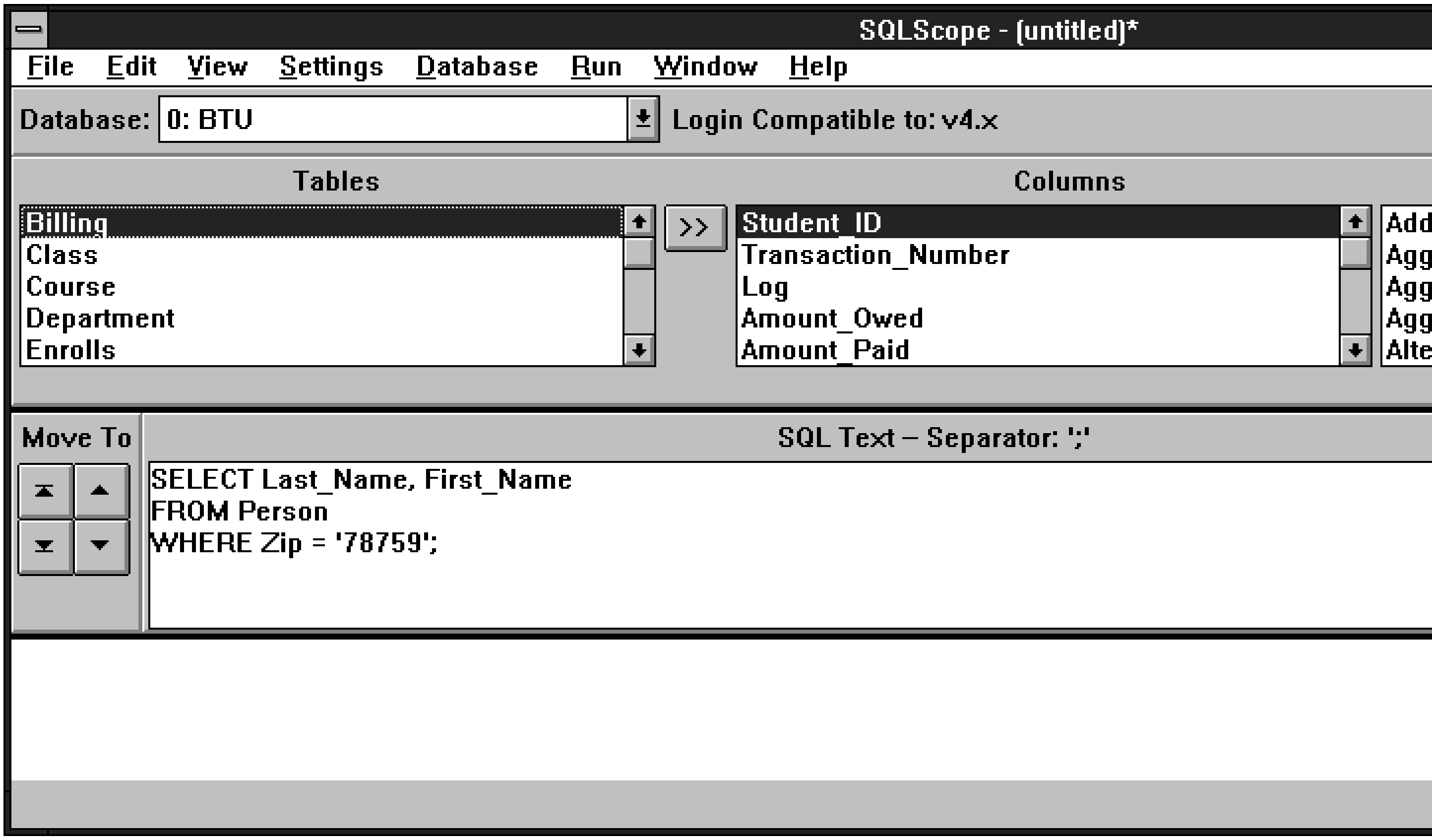
2. Choose **Logout** from the **Database** menu.

   You can also log out by choosing **Select** from the **Database** menu. In the **Select Database** dialog, highlight one or more databases and click the **Logout** button.

Creating and Running SQL Statements

The SQLScope main window (Figure 9-3) allows you to create SQL statements interactively.

Figure 9-3
SQLScope Main Window



Database: 0: BTU
Login Compatible to: v4.x

SQL Text – Separator: ';'

```
SELECT Last_Name, First_Name
FROM Person
WHERE Zip = '78759';
```

The main window allows you to do the following:

- View database information and select SQL templates.

  This area comprises the top of the SQLScope window. It contains the databases you are logged into, table names and column names from the currently selected database, and SQL statement templates you can use to build SQL statements. The update columns button (**>>**) allows you to update the columns list with the column names of the currently highlighted table.

- Compose, navigate, and run SQL statements.

  This area contains a text box for entering SQL statements and command buttons for moving among multiple SQL statements and running those statements.

- View results.

  This area displays the results of the statement you just executed.

- View status bar.

  This area displays a short description of the currently highlighted command or a progress report on current statement execution.

You can use your mouse or use commands on the **Window** menu to move among areas in the SQLScope window. The selected command reflects the active area. This area remains active until you move to another area.

Showing and Hiding Screen Elements

The **View** menu commands correspond to the following window areas:

Lists — Shows or hides the Database, Tables, Columns, and Templates lists.

Status Bar — Shows or hides the status bar.

Movement Buttons — Shows or hides the Move To buttons at the left of the SQL Text box.

Run Buttons — Shows or hides the Run buttons at the right of the SQL Text box.

Using Templates

The Templates list contains predefined templates of valid SQL syntax. Templates are not available in Version 3.01 Compatible mode.

To copy a template into the SQL Text box:

1. Scroll through the list to find the template you want or press the first letter of that template’s name.

2. When you have located the desired template, double-click the template name. (If multiple templates begin with the same letter, you can press the first letter of the template name multiple times to reach the template you want to use.)

   For example, to create a SELECT statement, double-click the Select template option. SQLScope inserts the following template into the SQL Text box at the text cursor:

```
SELECT [DISTINCT] < * | $Select_Terms_List >$Join_List
[$Where_Clause]
[$Group_By_Clause]
[$Order_By_Clause]
```

Note: Template names begin with a dollar sign ($). As the preceding example illustrates, template text may contain references to other templates. If you need help with a statement element, you can insert the referenced template for that element. For example, if you need help defining a WHERE clause, double-click on $Where_Clause to highlight that text. Then, double-click on the Where_Clause template in the Templates list box.

A second example sets column definitions in a CREATE TABLE statement using the Column_Def template:

```
#Column_Name #Data_Type ( #Data_Length ) [CASE]
```

> **Note:** Elements preceded by a pound sign (#) are user-defined variables that should be replaced with table names, column names, or constants. For example, you can replace the preceding example with values to produce a column definition such as ID CHAR (10). If you are using the pound sign as the statement separator, ensure that all extraneous instances of the pound sign are removed from your SQL script.

Creating SQL Statements

After logging in to a database, you can create and run SQL statements in the Statement Area. The following section provides information about how to create statements in SQLScope. For information about SQL statement syntax, refer to the *SQL Language Reference* .

You can create SQL statements in either of two ways:

- Enter the statement into the SQL Text box.
- Use the Tables, Columns, and Templates lists to assist you in creating the statement.

To use the Tables, Columns, and Templates lists to create a SQL statement:

1. Based on the type of statement you want to run, double-click a template from the Templates list.
2. In the SQL Text box, review the statement text and delete portions you do not want to use.

   The following SELECT statement includes only the required elements; all other data elements have been deleted.

```
SELECT $Select_Term$Join_List
```

3. Insert table and column names where appropriate.

   You can insert names by typing them directly into the SQL Text box, or you can double-click on them in the Tables or Columns list.

   To update the Columns list, select a table name from the Tables list and then either press the update columns button (**>>**), or choose **Update Columns List** from the **Database** menu.

> **Note:** If you select another table but do not update the Columns list, when you activate a window other than the Columns list box, SQLScope reselects the previously selected table.

4. If necessary, insert other values, such as constants.
5. *Optional :* Add comments at the end of any statement, if you wish.

   Comments begin with a delimiter (--) and end with a carriage return/line feed. You can place a comment on any line of your statement as long as it follows all of the statement text on that line.

The following example illustrates using comments. (A <CR> symbol indicates the end of a line.):

```
-- generate a list of people from Texas<CR>

SELECT Last_Name, First_Name, State<CR>Person<CR>State = ’TX’ -- limits the list to<CR>
    -- those who live in Texas<CR>
```

When Scalable SQL compiles the statement, it ignores all text between the comment delimiter and the end of the line (carriage return/line feed).

You can create multiple statements in the SQL Text box. Separate the statements with a statement separator; the semicolon (;) is the default. (See “Specifying the Statement Separator Character” for more information.) The following example creates two statements:

```
DECLARE BTUCursor CURSOR

FOR SELECT Degree, Residency, Cost_Per_CreditTuition ORDER BY ID;

OPEN BTUCursor;
```

Using the commands on the **Edit** menu, you can cut, copy, and paste text in the SQL Text box. In many cases, you can also undo the last edit you made.

Running SQL Statements

To run a SQL statement:

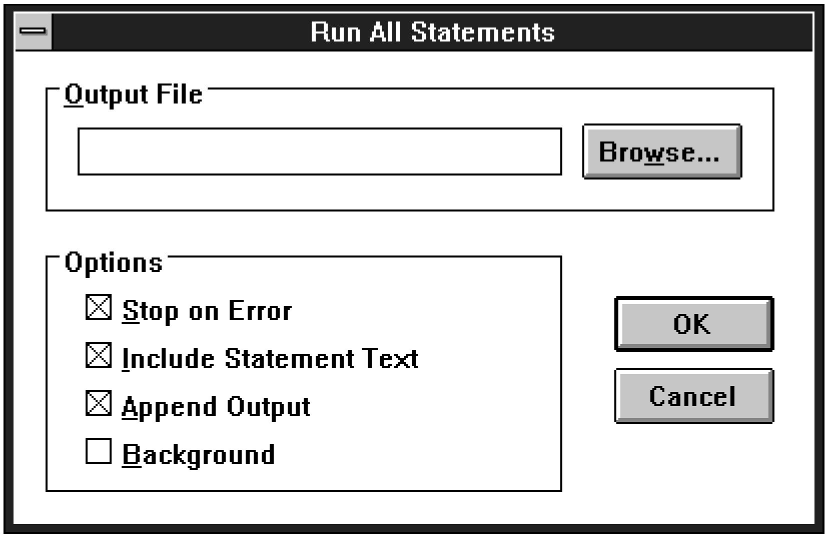
1. Position the text cursor in the SQL Text box on any character of the statement.
2. Then, click **Current** in the Statement Area or choose **Current** from the **Run** menu.

> **Note:** To run the first SQL statement in the SQL Text box, click **First** in the Statement Area, or choose **First** from the **Run** menu.

To run all statements in the SQL Text box:

1. Choose **All** from the **Run** menu. The **Run All Statements** dialog appears (Figure 9-4).

Figure 9-4
Run All Statements Dialog



☒ Stop on Error
☒ Include Statement Text
☒ Append Output
☐ Background

2. Using this dialog, you can redirect the results to an external file you specify. The **Run All Statements** dialog also allows you to select or clear the following check boxes:

Stop on Error — Stops executing the statements if SQLScope either encounters an error while accessing the external file or receives an error from Scalable SQL.

Include Statement Text — Includes the text of the statements as well as the results in the external file.

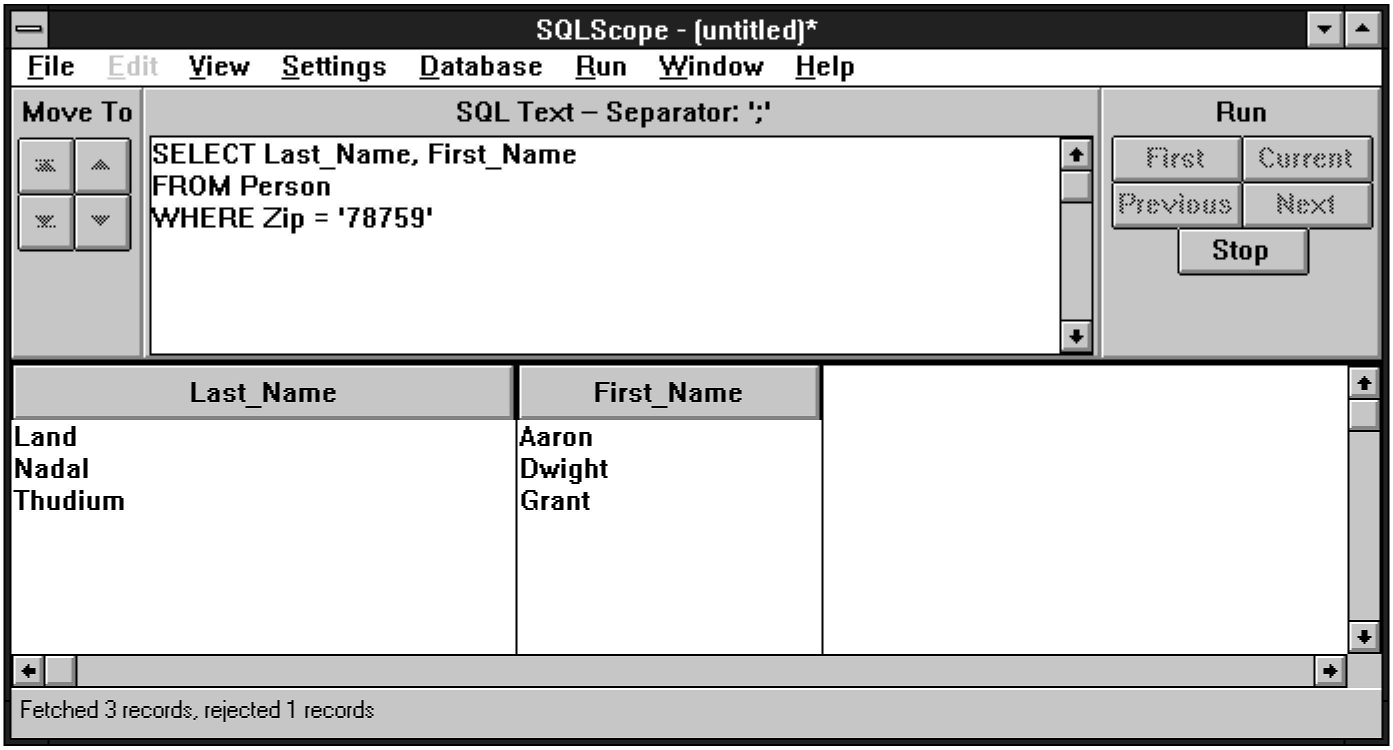
Append Output — Appends the results to an existing file. If you select the Include Statement Text option, this option includes the text of the statement in the output file, as well.

Background — Runs the statements in the background, allowing you to continue working in SQLScope.

> **Note: Background Mode:** You cannot exit SQLScope while a background process is running. Additionally, you cannot run SQL statements in the background connected to a Pervasive.SQL workstation engine. If you try to run in this mode, a Status Code 265, “The session identifier is invalid,” occurs. You *can* run in this mode when you are connected to a Pervasive.SQL server engine and the SQL Requester Thunk setting is configured to *No* .

When executing statements, SQLScope hides the Database, Tables, Columns, and Templates lists, enters Run mode, and does not allow you to change the SQL statements. (You can still scroll through existing SQL statements in the SQL Text box.) Figure 9-5 shows the SQLScope main window in Run mode.

Figure 9-5
SQLScope Main Window in Run Mode



```
SELECT Last_Name, First_Name
FROM Person
WHERE Zip = '78759'
```

Last_Name	First_Name
Land	Aaron
Nadal	Dwight
Thudium	Grant

Fetched 3 records, rejected 1 records

While SQLScope is in Run mode, if you specified multiple statements, you can also use one of the following Run buttons in the Statement Area or commands from the **Run** menu:

Previous Runs the statement that precedes the current one.

Next Runs the statement that follows the current one.

Results appear in the Results Area, as follows:

- For a SELECT statement, SQLScope displays the result table. If the result table is larger than the Results Area, you can scroll through the data. Also, you can adjust the width of any column. For more information, refer to “Adjusting Column Widths in the Results Area” .

- For statements other than SELECT, SQLScope displays the status of the statement execution.

After reviewing the Results Area, you can either run another statement or exit Run mode:

- To remain in Run mode and run another statement from the SQL Text box, click the **Previous** or **Next** button.

- To exit Run mode, click the **Stop** button or choose the **Stop** command on the **Run** menu. The hidden lists reappear, and SQLScope allows you to edit statements in the SQL Text box.

Saving SQL Scripts

If you use a statement or group of statements often, you can save the statement or statements in a script file for later use. The script file is a text file you can edit using any standard ASCII text editor. Script files cannot exceed 32 KB in size.

Note: Do not confuse scripts you save in SQLScope with stored procedures. Scalable SQL precompiles and saves stored procedures in the X$Proc table, whereas SQLScope saves SQL scripts to a script file. SQLScope does not precompile these statements.

To create a new script file and save it:

1. Create one or more statements as described in "Creating and Running SQL Statements".

2. Choose **Save** from the **File** menu.

   SQLScope displays the **Save SQL Script File** dialog.

3. Enter a full path—including drive, directory, and file name.

To recall an existing script file:

1. Choose **Open** from the **File** menu.

2. In the **Open SQL Script File** dialog, enter the full path of the script file you want to recall.

   SQLScope opens the script file and displays it in the SQL Text box. In addition, SQLScope displays the full path of the script file in the title bar.

3. If you make changes to an existing script file, you can save the changes by choosing **Save** from the **File** menu. SQLScope saves your changes and does not display the **Save SQL Script File** dialog.

To rename an existing script file, follow these steps:

1. Open an existing script file.

2. Choose **Save As** from the **File** menu.

   SQLScope displays the **Save SQL Script File** dialog.

3. Enter a full path—including drive, directory, and file name.

Adjusting Column Widths in the Results Area

Once you run a SQL statement and SQLScope displays the results in the Results Area, you can adjust the column widths. Adjusting the column widths affects their display only for the current statement execution. In other words, if you run another statement, the columns revert to their default display widths.

SQLScope provides two methods for adjusting column widths:

- Direct manipulation using the mouse cursor

- Numeric settings using **Column Widths** from the **Settings** menu

To adjust column widths using direct manipulation:

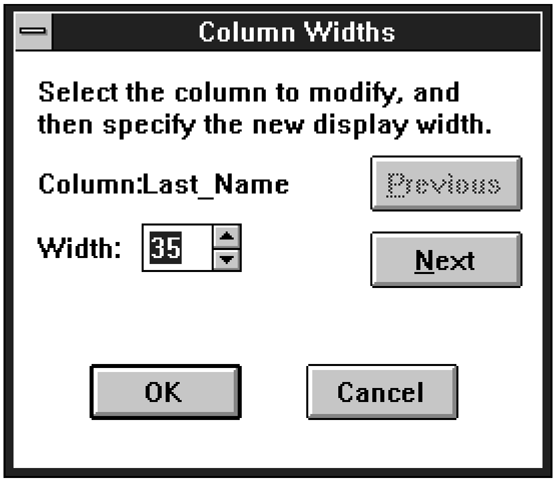
1. Place the cursor on the margin between two column headings.
2. When the cursor changes to a vertical bar with arrows, you can drag and drop the margin to a new location.

To adjust column widths using the Column Widths command:

1. Choose **Column Widths** from the **Settings** menu.

   SQLScope displays the **Column Widths** dialog (Figure 9-6).

Figure 9-6
Column Widths Dialog



Column Widths

Select the column to modify, and then specify the new display width.

Column:Last_Name

Width: 35

2. If necessary, use the **Next** or **Previous** button to move to another column.
3. Use the Width text box to specify a new width for the column.

   Because SQLScope displays results using proportionally spaced text, the Width value reflects an approximate number of characters.
4. Click **OK** to exit, or perform Steps 2 and 3 for another column.

Updating the Lists Area

You can use the **Refresh Lists** command on the **Database** menu to reread all the database table names from the data dictionary and display them in the Tables list. SQLScope maintains highlighting on a table if you highlighted the table before you chose the **Refresh Lists** command. SQLScope also refreshes the Columns list, if necessary.

Note: This command is useful after you execute a series of SQL statements that change the tables or columns in the current data dictionary.

Managing Referential Integrity

> **Note:** SQLScope's RI commands work on Windows NT servers only. If you want to check the RI of a database on a NetWare server, refer to "Checking and Repairing Referential Integrity".

You can use SQLScope to do the following:

- Generate a report about the referential constraints defined for the database.
- Verify or repair RI on the database. You should verify RI when you add referential constraints to tables or restore a partial backup of a database.

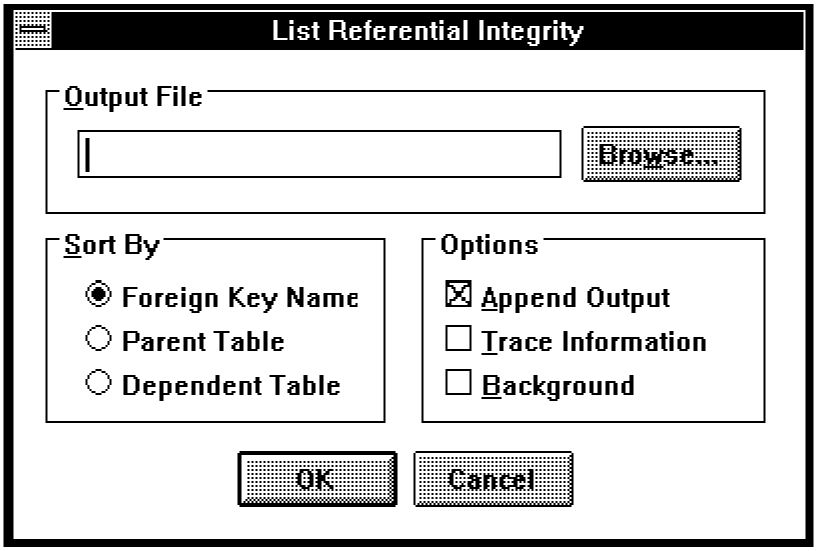
You must log into the database using its database name. Your database must be named, stored on the same Windows NT server as the SQL Engine, and have referential constraints defined (whether or not RI is enabled). Also, you must use the 16-bit version of the Setup utility to enable the Use Thunk option for both the SQL Engine and the MicroKernel Router.

Listing Referential Constraints

To produce a report that lists all foreign key definitions and related information for any named database:

1. Log in to the database using its database name.
2. Choose **List Constraint** s from the Database menu.
3. In the Output File text box of the List Referential Integrity dialog box (Figure 9-7), enter the full path of the report file to which you want SQLScope to write. If the file does not exist, SQLScope creates it.

Figure 9-7
List Referential Integrity Dialog



List Referential Integrity

Output File

Sort By

● Foreign Key Name
○ Parent Table
○ Dependent Table

Options

☒ Append Output
☐ Trace Information
☐ Background

4. *Optional* : By default, SQLScope appends output to existing files. If the output file you specified exists and you want SQLScope to write over the existing file contents, deselect the Append Output check box.

5. *Optional* : If you want the report to include additional trace information (such as header information about the options you specified, full paths, and detailed information about each table in the report), select the Trace Information check box.

6. *Optional* : Select a sort order for the information presented in the report.

The Sort By options are as follows:

Foreign Key Name	Sorts the output by the names of the foreign keys defined in the database.
Parent Table	Sorts the output by the names of the parent tables defined in the database.
Dependent Table	Sorts the output by the names of the dependent tables defined in the database.

By default, SQLScope sorts the information by foreign key name.

7. *Optional* : To generate the report in the background so you can continue working in SQLScope, select the Background check box.

Note: You cannot exit SQLScope while a background process is running.

8. Choose **OK** or press enter.

SQLScope generates the report, writes it to the output file you specified, and displays the Listing Referential Constraints dialog box.

9. If you want to view the report file, choose **View** . If an error occurred, you can choose **Status** to display the status code and error message.

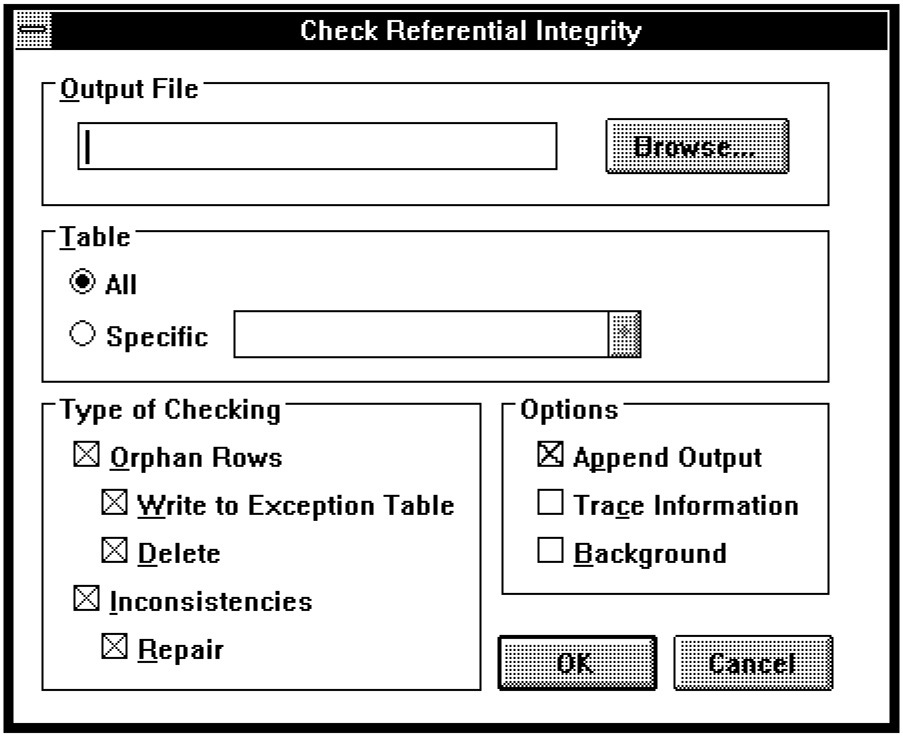
Checking Referential Integrity

To check for orphan rows in a named database or verify the consistency of a data file's RI data with the data dictionary's RI data:

1. Log in to the database using its database name.
2. Choose **Check Constraints** from the **Database** menu.
3. In the **Output File** text box of the **Check Referential Integrity** dialog box (Figure 9-8), specify the report file to which you want SQLScope to write.

   To specify an output file, you can either enter the full path of the file or choose **Browse** and choose a path. If the file does not exist, SQLScope creates it.

Figure 9-8
Check Referential Integrity Dialog



Check Referential Integrity

Output File

Table
● All
○ Specific

Type of Checking
☒ Orphan Rows
☒ Write to Exception Table
☒ Delete
☒ Inconsistencies
☒ Repair

Options
☒ Append Output
□ Trace Information
□ Background

4. *Optional* : By default, SQLScope appends output to existing files. If the output file you specified exists and you want SQLScope to write over the existing file contents, deselect the **Append Output** check box.
5. *Optional* : If you want the report to include additional trace information (such as header information about the options you specified, full paths, and detailed information about each table in the report), select the Trace Information check box.
6. *Optional* : If you want to check only one table in the database, select the Specific option button and specify a table name.

Otherwise, SQLScope checks and reports on all tables in the database.

7. *Optional* : By default, SQLScope checks for orphan rows, writes them to an exception table, and deletes them.

   You can change the relevant settings as follows:

Orphan Rows	If you do not want SQLScope to check for orphan rows, deselect this option button.
Write to Exception Table	If you do not want SQLScope to create an exception table, deselect this option button.
Delete	If you do not want SQLScope to delete the orphan rows it finds, deselect this option button.

   For more information, refer to “Exception Tables”.

8. *Optional* : By default, SQLScope checks for and repairs any inconsistencies between the contents of the data files and the referential constraints contained in the data dictionary.

   You can change the relevant settings as follows:

Inconsistencies	If you do not want SQLScope to check for inconsistencies, deselect this option button.
Repair	If you do not want SQLScope to repair inconsistencies, deselect this option button.

   For more information, refer to “Database Inconsistencies”.

9. Choose **OK** or press enter.

   SQLScope checks the database’s RI, generates the report, writes it to the output file you specified, and displays the Checking Referential Constraints dialog box. If you want to view the report file, choose **View** . If an error occurred, you can choose the Status button to display the status code and error message.

Exception Tables

By default, SQLScope generates an exception table for each table in which it finds orphan rows. The exception table becomes part of the database, with the same location and file name as the original data file, but with a .EXC extension. For example, if SQLScope generated an exception table on the Patients table, the exception table would be named EXC_Patients and would be stored in a data file named PATIENTS.EXC.

The first field in the exception table is an index and contains the parent table name. The remainder of each row contains the same field values as the original orphan row and can contain up to 4,090 bytes. SQLScope truncates rows larger than 4,090 bytes. You can use SQLScope to review the exception table by issuing SQL statements, just as you would with any other table.

Database Inconsistencies

By default, SQLScope checks for inconsistencies between the information in the data dictionary and that in the individual data files. For example, you may create an inconsistency if you move a data file from one database to another, because the old database name stored in the data file does not match the new database name stored in the data dictionary. SQLScope checks for the following inconsistencies:

Database Names	Checks the database name stored in the data file against the database name stored in the data

	dictionary.
Primary Key	Checks the number of referencing foreign keys stored in the data file against the referential constraints stored in the data dictionary.
Foreign Keys	Checks the number of foreign keys defined in the data file against the referential constraints stored in the data dictionary.

To repair inconsistencies, SQLScope updates the information stored in the individual data files to match that in the data dictionary.

Reviewing File-Level Information

To display file-level information about Scalable SQL data files:

1. Log in to a database.

2. In the Tables list, highlight the table for which you want to see data file information.

3. Choose **Table Statistics** from the **Database** menu. The **Table Statistics** dialog appears, as shown in Figure 9-9.

   You can only view the statistics SQLScope displays; you cannot change them. Table 9-2 describes the information SQLScope displays in the **Table Statistics** *dialog.*

Figure 9-9
Table Statistics Dialog

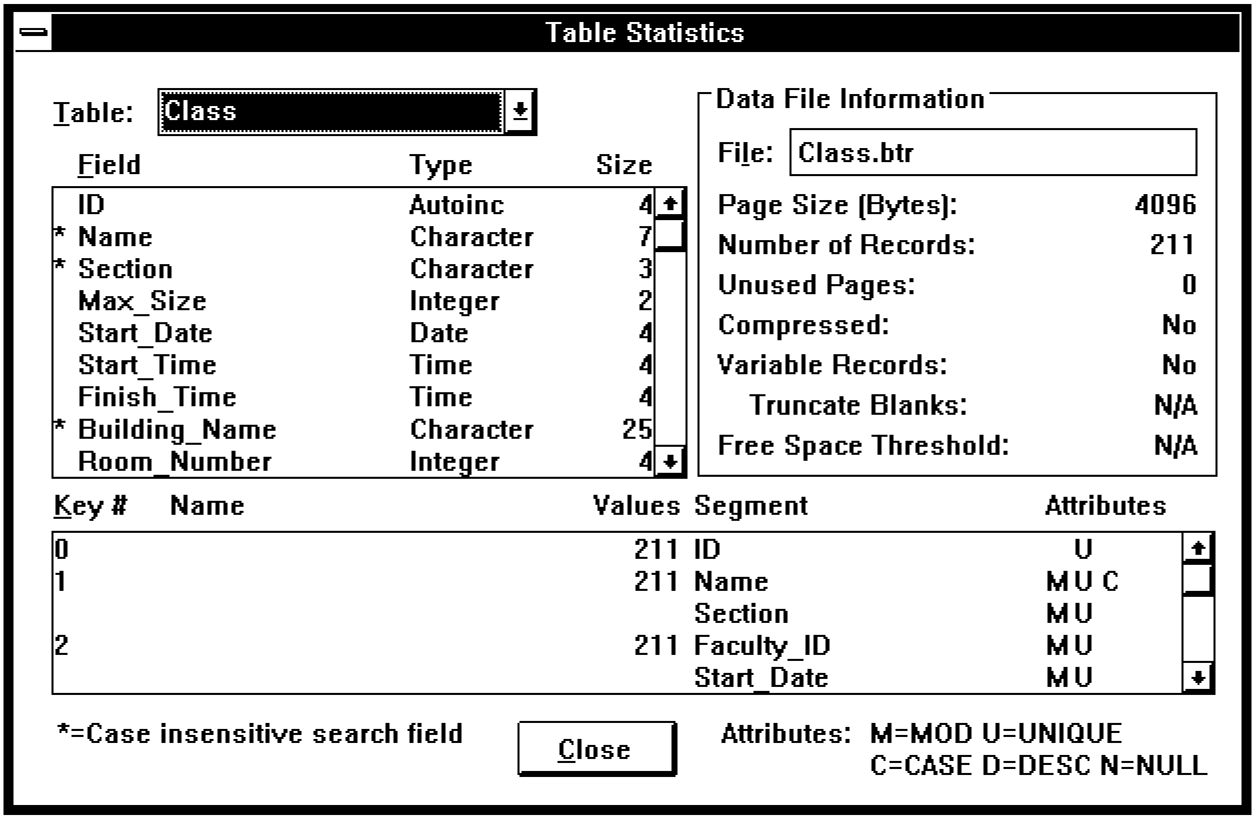


Table: Class

Field	Type	Size
ID	Autoinc	4
* Name	Character	7
* Section	Character	3
Max_Size	Integer	2
Start_Date	Date	4
Start_Time	Time	4
Finish_Time	Time	4
* Building_Name	Character	25
Room_Number	Integer	4

Data File Information

File: Class.btr

Page Size (Bytes):	4096
Number of Records:	211
Unused Pages:	0
Compressed:	No
Variable Records:	No
Truncate Blanks:	N/A
Free Space Threshold:	N/A

Key #	Name	Values	Segment	Attributes
0		211	ID	U
1		211	Name	M U C
			Section	M U
2		211	Faculty_ID	M U
			Start_Date	M U

*=Case insensitive search field

Attributes: M=MOD U=UNIQUE
C=CASE D=DESC N=NULL

4. *Optional:* To display statistics for another table, use the Table drop-down list to select that table.

5. When you finish reviewing table statistics, click **Close** .

The following table describes the information SQLScope displays in the **Table Statistics** dialog.

Table 9-2
Table Statistics and Descriptions

Statistic Name	**Description**
Column	Lists the columns defined for the current table.
Type	Shows the data type defined for each column.
Size	Shows the size in bytes of each column.
File	Shows the data file associated with the current table.
Page Size	Shows the page size (in bytes) of your data file. The page size determines the maximum number of index segments you can define in the table.Scalable SQL uses a default page size of 4096 bytes when creating data files. When you use a CREATE TABLE statement, you can specify a page size other than the default.
Number of Records	Shows the number of records the data file contains.
Unused Pages	Shows the number of preallocated pages available. If preallocation is enabled, the MicroKernel preallocates a specified number of pages when it creates the data file. Preallocation guarantees that disk space for a data file is available when the MicroKernel needs it.When you use a CREATE TABLE statement, you can enable preallocation and specify the number of pages to preallocate.
Compressed	Shows whether data compression is enabled. If it is, the MicroKernel compresses each record it inserts into the data file. When you use a CREATE TABLE statement, you can enable compression.
Variable Records	Shows whether the data file contains variable-length records.
Truncate Blanks	Shows whether blank truncation is enabled. If it is, the MicroKernel truncates the blanks in variable-length records. Blank truncation is applicable only if the Variable Records statistic is Yes and Data Compression is set to No.
Free Space Threshold	Displays a percentage (5%, 10%, 20%, or 30%) if the data file has a free space threshold. The MicroKernel stores the variable-length portions of records on their own pages (called variable pages), separate from the fixed-length portions (which are stored on data pages).The MicroKernel uses the threshold to determine whether to add data to an existing variable page or to create a new one. A higher free space threshold reduces fragmentation of variable-length records across several pages but uses more disk space.The threshold is applicable only if the Compressed or Variable Records statistic is Yes. When you use a CREATE TABLE statement, you can specify a free space threshold.
Key #	Lists the key numbers for the current table. The MicroKernel stores Scalable SQL indexes as keys.
Name	Displays the name of the index, if applicable. For more information about named indexes, refer to the *Pervasive.SQL Programmer's Guide* .
Values	Displays the number of column values stored for the index.
Segment	Displays the column or columns on which the index is defined. For more information about indexes and index segments, refer to the *Pervasive.SQL* Programmer's Guide.
Attributes	Displays the attributes defined for the index. For more information, refer to the *Pervasive.SQL Programmer's Guide* .

Importing and Exporting Data

SQLScope can import and export data stored in the UNF, SDF, or ASCII data formats. For more information about these formats and the rules for importing and exporting data, refer to Table 7-2. For information about the role of data types and defined or default masks, refer to the *SQL Language* Reference.

Importing Data

You can use the **Import** command on the **File** menu to insert, update, or delete data in an existing Scalable SQL database. The data you insert, update, or delete must be specified in a data file format discussed in Table 7-2. The import file must contain the same number of columns referenced by the SQL statement.

To import data from a file in one of the supported file formats:

1. In the SQL Text box, for SDF or ASC data formats create an INSERT, UPDATE, or DELETE statement containing a substitution variable for each column in the import file. For the UNF format, create a SELECT statement that contains no substitution variables.

   The following example inserts rows into the Course table in the sample database. In this example, the import file contains values for the Name, Description, and Credit_Hours columns.

```
INSERT INTO Course
(Name, Description, Credit_Hours)(@V1, @V2, @V3)
```

   The following example deletes rows from the Course table in the sample database. In this example, the SDF or ASC import file contains values for the Name column.

```
DELETE FROM CourseName = @V1
```

   The following example selects the course name, description, and credit hours from the Course table. In this example, the import file must be in UNF format.

```
SELECT Name, Description, Credit_HoursCourse
```

   For more information about substitution variables, refer to the *SQL* Language Reference.

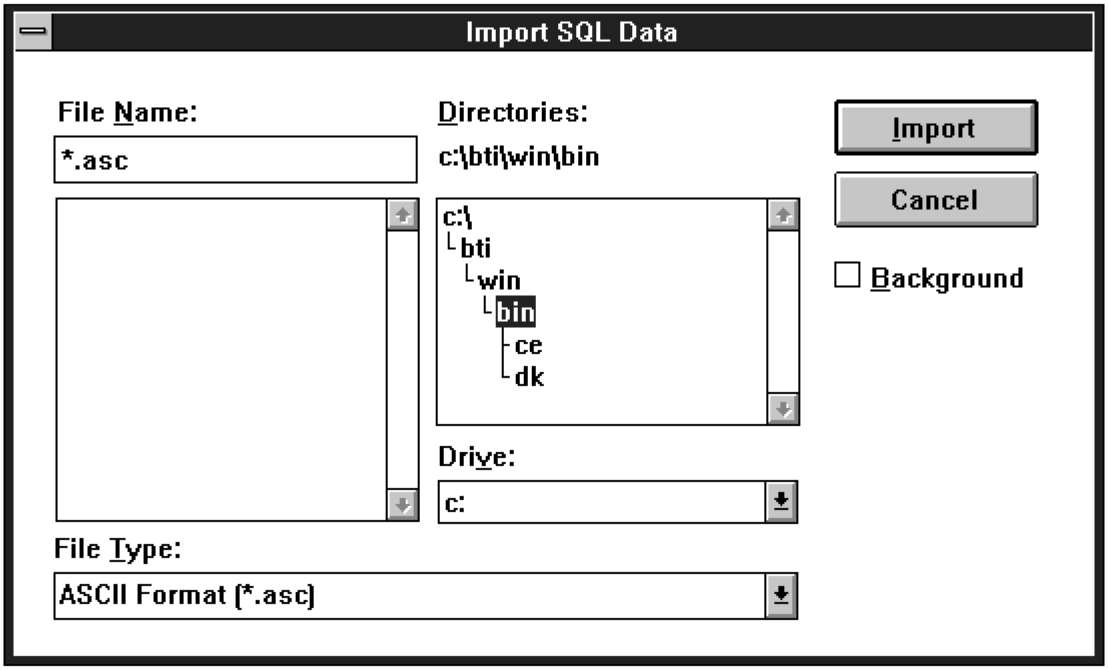
2. Choose **Import** from the **File** menu.

3. In the **Import SQL Data** dialog (Figure 9-10), specify the following:

   - In the File Type drop-down list, specify the type of the file to import.

   - Specify the full path of the file in the **Directories** box and the **File Name** text box.

Figure 9-10
Import SQL Data Dialog



4. *Optional* : To import the file in the background so that you can continue working in SQLScope, select the **Background** check box.

Note: Background Mode: You cannot exit SQLScope while a background process is running. Additionally, you cannot run SQL statements in the background if you are connected to a Pervasive.SQL workstation engine. If you try to run in this mode, a Status Code 265, “The session identifier is invalid,” occurs. You *can* run in this mode when you are connected to a Pervasive.SQL server engine and the SQL Requester Thunk setting is configured to *No* .

5. Click **Import** or press enter.

   Using the SQL statement you specified, SQLScope imports data from the specified file into the current database. While importing data, SQLScope displays the **SQLScope Import** dialog, which shows the path of the import file and the number of rows imported. You can pause the import by clicking **Pause** ; you can stop the import by clicking **Stop** .

Note: Once you have started an import operation, you cannot cancel it and return the database to its state before the import.

6. When you are finished importing data, click **Close** .

Exporting Data

To export data to a file in one of the supported file formats:

1. In the SQL Text box, create a SELECT statement that selects all the columns to include in the exported file. SQLScope exports the data in its current mask or in the default mask if a mask does not exist.

   For more information about creating statements using SQLScope, refer to "Creating and Running SQL Statements." For examples of SELECT statements, refer to "SAVE". For more information about SQL statement syntax, refer to the *SQL Language* Reference.

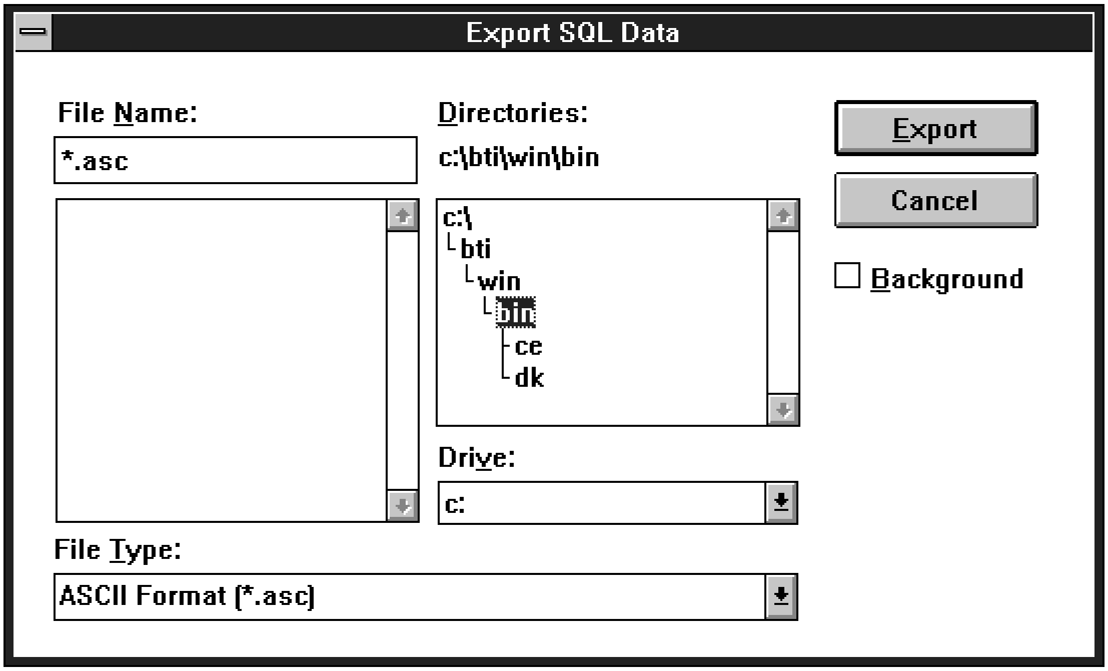
2. Choose **Export** from the **File** menu.

3. In the **Export SQL Data** dialog (Figure 9-11), specify the following:

   - In the **File Type** drop-down list, specify the type of the file you want SQLScope to create. Refer to Table 8-2 for more information about file types.

   - In the **File Name** text box, specify the full path of the file that will contain the exported data.

Figure 9-11
Export SQL Data Dialog



Export SQL Data

File Name: *.asc

Directories: c:\bti\win\bin

c:\
└bti
 └win
  └bin
   ├ce
   └dk

Drive: c:

File Type: ASCII Format (*.asc)

4. *Optional* : To export the file in the background so that you can continue working in SQLScope, select the **Background** check box.

Note: Background Mode: You cannot exit SQLScope while a background process is running. Additionally, you cannot run SQL statements in the background if you are connected to a Pervasive.SQL workstation engine. If you try to run in this mode, a Status Code 265, "The session identifier is invalid," occurs. You *can* run in this mode when you are connected to a Pervasive.SQL server engine and the SQL Requester Thunk setting is configured to *No* .

5. Click **Export** or press enter.

   Using the SQL statement you specified, SQLScope exports data from the current database into the specified file. While exporting data, SQLScope displays the **SQLScope Export** dialog, which shows the path of the export file and the number of rows exported. You can pause the export by clicking **Pause** ; you can stop the export by clicking **Stop** .

> **Note:** Once you have started an export operation, you cannot cancel it and return the database to its state before the export.

6. When you are finished exporting data, click **Close** .

Recovering Damaged Data Files

To recover a SQL Interface data file with damaged index information:

1. Log in to the database containing the damaged file.

For more information about logging in to databases, refer to “Logging in to a Database”.

2. In the SQL Text box, create a SELECT statement that selects all the columns in the damaged file.

The following example selects all columns from the Course table:

```
SELECT * FROM Course
```

> **Note:** Do not use a WHERE clause in the SELECT statement. Doing so may cause Scalable SQL to read by an index. (The SQL Interface reads only data pages when you use SELECT statements without WHERE clauses.)

For more information about creating statements using SQLScope, refer to “Creating and Running SQL Statements”. For more information about SQL statement syntax, refer to the *SQL Language* Reference.

3. Choose **Export** from the **File** menu to export the data to a UNF file.

For more information about exporting data, refer to “Exporting Data”.

4. In the SQL Text box, execute an ALTER TABLE statement that replaces the existing data file.

The following example replaces the data file for the Tuition table:

```
ALTER TABLE Course’course.mkd’REPLACE
```

> **Note:** Be sure to specify the existing data file name. Doing so ensures that Scalable SQL replaces the damaged file.

For more information about ALTER TABLE statements, refer to the *SQL* Language Reference.

5. In the SQL Text box, create a SELECT statement with all the columns in the data file. The following example inserts data into the Student table:

```
SELECT (ID, Cumulative_GPA, Tuition_ID, Transfer_Credits, Major,
Minor, Scholarship_Amount, Cumulative_Hours)(@id, @cumulative_gpa, @tuition_id, @transfer_credits,
@major, @minor, @scholarship_amount, @cumulative_hours)Students
```

6. Use the Import command from the File menu to import the data from the UNF file you created in Step 3.

For more information about importing data, refer to “Importing Data”.

Customizing SQLScope

You can customize the following items in SQLScope:

- Default login values
- Environment settings, such as default values in dialogs, screen layout, statement separator characters, and options for saving the settings

Specifying Default Login Settings

You can customize the **Database Login** dialog to automatically include a database name or location, set a username, and set the compatibility mode. The next time you log in, your defaults will be in effect.

To set default login settings:

1. Choose **Login Defaults** from the **Settings** menu.
2. In the **Database Login Default Settings** dialog, enter a database name or data dictionary location.

   If the database requires a username, you can also specify your username. For security reasons, SQLScope does not provide a way to save your password.

   You can also specify the default compatibility mode. When you select the Version 3.01 Compatible check box, you can work in Scalable SQL 3.01 databases and ensure that they remain compatible with 3.01 applications. For more information about the Version 3.01 Compatible check box, refer to page 9-6.
3. Click **OK** to save your changes, or click **Cancel** to exit without saving any changes.

The preceding steps save the login settings for your current SQLScope session only.

To save the settings for future SQLScope sessions:

1. Choose **Save** from the **Settings** menu.
2. Select the Login check box in the **Save Settings** dialog.
3. Click **Save** to save your changes, or click **Cancel** to exit without saving any changes.

Specifying Environment Settings

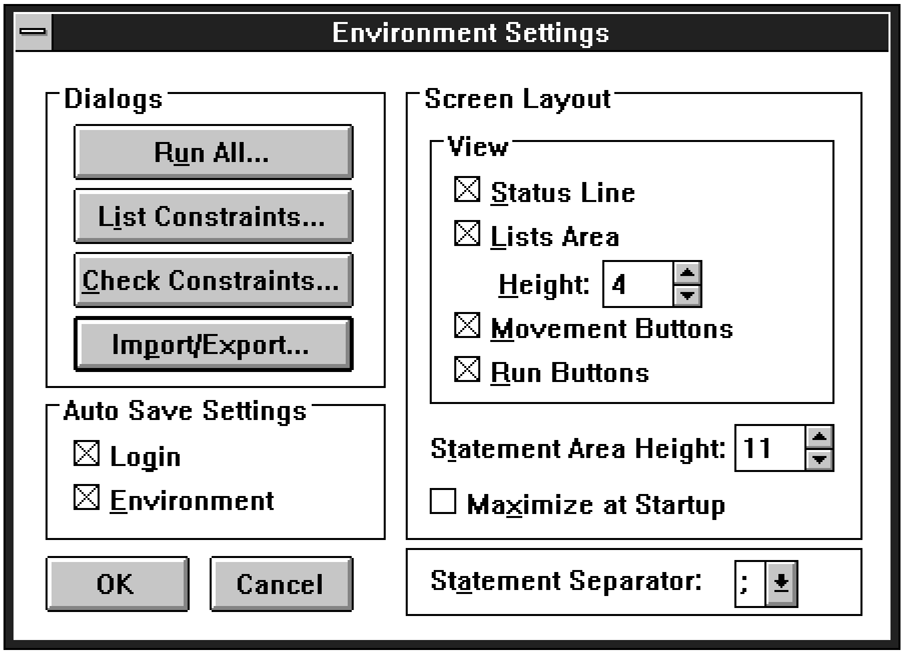
You can customize SQLScope to automatically include certain values in dialog boxes and to alter screen layout. You can also set up SQLScope to automatically save the settings for future SQLScope sessions.

To specify environment settings:

- Choose **Environment** from the **Settings** menu. The **Environment Settings** dialog appears (Figure 9-12).

Figure 9-12

Environment Settings Dialog



Auto Save Settings

☒ Login

☒ Environment

Screen Layout

View

☒ Status Line

☒ Lists Area

Height: 4

☒ Movement Buttons

☒ Run Buttons

Statement Area Height: 11

☐ Maximize at Startup

Statement Separator: ;

Setting Default Values for Dialogs

You can set default values for the **Run All Statements** and the **Import SQL Data** and **Export SQL Data** dialogs.

To set a default value for any of these dialogs:

- Click the corresponding button in the Dialogs box and enter the values.

For information about the options in each dialog, refer to the following:

Run All	Figure 9-4
Import/Export SQL Data	Figure 9-10

Automatically Saving Settings

To automatically save the settings you specify during your current SQLScope session for use in future SQLScope sessions:

- Use the check boxes in the Auto Save Settings box.

If you select a check box, SQLScope saves the appropriate settings and uses them in subsequent SQLScope sessions.

Setting the Default Screen Layout

You can specify some basic defaults for the SQLScope window layout in the View box in the **Environment Settings** dialog.

To show or hide screen elements:

1. Use the check boxes in the View box.

2. If you select a check box, that element appears on the screen. If you clear a check box, SQLScope hides that screen element. You can show or hide the following screen elements:

Status Bar Shows or hides the Status Bar at the bottom of the screen.

Lists Area Shows or hides the Tables, Columns, and Templates drop-down lists.

Movement Buttons Shows or hides the Move To buttons at the left of the SQL Text box.

Run Buttons Shows or hides the Run buttons at the right of the SQL Text box.

3. To change the height of the **Lists Area** , enter a value in the **Height** text box. The default height is 4 lines.

4. To change the height of the SQL Text window, enter a value in the **Statement Area Height** text box. The number you enter sets the number of text lines SQLScope displays at one time. The default height is 11 lines.

5. If you want the SQLScope window to automatically expand to the maximum window size for your screen in subsequent SQLScope sessions, select the **Maximize at Startup** check box.

Specifying the Statement Separator Character

You can specify the character to use to separate SQL statements. The default separator is a semicolon (;). However, because the syntax of stored procedure declarations include a semicolon, you can specify a different character to separate your SQL statements.

To change the statement separator character:

1. Select a new character from the Statement Separator drop-down list.

2. Choose either the pound sign (#) or a semicolon (;).

Checking and Repairing Referential Integrity

Note: *Windows NT Users and Workstation Engine Users* : You can use the RI commands in SQLScope to perform the operations of the RI Utility, which are described in “Managing Referential Integrity” in Chapter 8.

The Referential Integrity (RI) utility is a command-line utility that runs as an NLM at the file server. You can access it at the file server console or through the RCONSOLE remote file server console utility. Refer to the RI Utility Overview for details on running the utility and for conceptual information you need to know before using the RI utility.

This chapter includes the following sections:

- “RI Utility Overview”
- “Verifying and Re-establishing RI on a Named Database”

Note: This utility works only on named databases with referential constraints defined. For more information about referential integrity and referential constraints, refer to the *Pervasive.SQL Programmer’s Guide* .

RI Utility Overview

To run the RI utility:

- Enter the following command at the file server console prompt:

```
RIUTIL [ -command [parameter ...] ] | @file
```

command An RIUTIL command, as follows:

LIST:

Lists all foreign key definitions and related information.

CHECK:

Verifies the consistency of data files and checks for orphan records.

parameter Information the RI utility may need to perform the command you enter. The detailed description of each RI utility command (in the following sections) includes a discussion of the parameters it requires.

file Full path of a command file.

To view a brief explanation of commands:

- Enter the following command:

```
riutil
```

> **Note:** You cannot run the RIUTIL CHECK command on a database in which any users have files open. However, you can run the RIUTIL LIST command on such a database. If you run the RIUTIL LIST command with users logged in, any changes they make do not appear in the LIST reports.

RI Utility Command Files

You can use command files to do the following:

- Execute commands that are too long to fit on the command line.
- Enter commands that you use often. These command files contain the same information required on the command line.

Command File Rules

Observe the following rules when creating a command file:

- Limit each line to 130 characters.

Note: Lines longer than 130 characters could cause the file server to abnormally end. For this reason, do not place long RIUTIL commands in a file server command (.NCF) file.

- Do not split a single parameter across two lines.
- Use no more than one command per command file.
- Limit the command file size to no more than 1,000 bytes.

Example

The following is an example command file, LIST.CMD. It calls the RIUTIL –LIST command to list all foreign key definitions and related information for the BTU named database. It redirects the output to the INFO.TXT file.

```
-list
btu
/O:sys:\SSQL\demodata\info.txt
```

The following command uses the LIST.CMD file.

```
riutil @sys:\ssql\demodata\list.cmd
```

RI Utility Commands

The following paragraphs describe concepts you should understand before using the RI utility.

Named Databases

The RI utility works with named databases only; it relies on information stored in the data dictionary to find data files. When scanning the dictionary, the utility accepts only the following path formats:

vol : dir\ file	Absolute path
file or *dir* *file*	Relative path

An *absolute path* specifies the exact location of the file. Absolute path must specify a volume.

A *relative path* is a file name or path that is appended to each data file location associated with the named database, until the file is found. Suppose the database named BTU has two data file locations defined in this order: \SSQL\DEMODATA1 and \SSQL\DEMODATA2.

If you specify a relative file name such as FACULTY.MKD, the utility searches for that file first in the \SSQL\DEMODATA1 directory. If the utility does not find the file, it searches next in the \SSQL\DEMODATA2 directory.

Note: Because the server environment in which RIUTIL operates does not support the use of drive letters or implied drives, specifying either one results in a Status Code 11. The only instance in which the RI utility uses an assumed volume is when the file resides on the SYS: volume.

Referential Constraints

Referential integrity is the assurance that when a column (or group of columns) in one table references a column (or group of columns) in another table, changes to these columns are synchronized. Referential constraints are the rules that define the relationships between tables.

This discussion of the RI utility includes references to the following terms:

Dependent table	A table containing at least one foreign key.
Foreign key	A column or group of columns that reference a primary key in the same or a different table.
Orphan record	A record whose foreign key value does not have a matching value in its parent table.
Parent table	A table containing a primary key that is referenced by foreign keys in dependent tables.
RI data	Information on referential constraints used internally by the SQL Interface and stored in both the data files and the dictionary files.

Verifying and Re-establishing RI on a Named Database

This section includes the following functions:

- “Generating a Referential Constraints Report Using LIST”
- “Checking for Orphan Records and Repairing Inconsistent RI Information”
- “Generating a List of Orphan Records or Inconsistencies”
- “Generating Exception Tables”

LIST

The LIST command lists all foreign key definitions and related information for any named database on the current server.

Format

```
RIUTIL -LIST [database_name]/O:vol:path   [option...]
```

database_name The name of the database for which you want to see referential integrity information. If you enter an asterisk (*) for this parameter, the RI utility produces referential integrity information for all named databases on the server.

/O: vol: path The full path of the file in which you want the RI utility to place the generated information.

For *option* , you can specify any of the following options in any order:

/P: password The master password of dictionary files associated with the specified named database. The SQL Interface requires a password if security is enabled for the database. If you specify the /P: option followed by an asterisk (*), the RI utility prompts you for each password as it opens the database.

/A Appends the generated information to an existing file, if you specify an existing file with the /O: option.

/S: sort The order in which you want information listed in the report. Specify one of the following:

N:
Foreign key name

P:
Parent table

D:
Dependent table

/T Displays additional trace information. This option is helpful in diagnosing database problems.

/? Displays the syntax for the LIST command.

If you do not specify a database name, the LIST command shows all available named databases, as in the following example.

```
Database Names on This Server
BTU
BTUDEMO
```

The command also displays a short usage message.

Generating a Referential Constraints Report Using LIST

To place the report information in a text file:

1. Use the LIST command and specify the /O: option as in the following example:

```
RIUTIL -LIST btu /O:sys:\ssql\demodata\info.txt
```

The output file contains information about the relationships between tables in a database upon which referential constraints are enforced.

For example, the LIST command might generate the following output (see Example 10-1) on the BTU database.

Example 10-1
Referential Constraints Report

```
      Foreign Key          : BILLINGBYREGISTRAR

Parent Table               : Person                      Key #: 0     Column(s): ID



Dependent Table            : BILLING                     Key #: 2     Column(s):
                                                                      REGISTRAR_ID

*****************

Delete Rule                : Restrict

Update Rule                : Restrict



Foreign Key                : BILLINGFORASTUDENT

Parent Table               : STUDENT                     Key #: 0     Column(s): ID



Dependent Table            BILLING                       Key #: 3     Column(s):
                                                                      STUDENT_ID
```

Delete Rule	Restrict
Update Rule	Restrict

2. By default, the RI utility sorts the information by foreign key name. Use the /S:P option to group the foreign keys by parent table name or the /S:D option to group the foreign keys by dependent table name.

3. By default, the RI utility overwrites any existing information in the output file. Specify the /A option to append new information to the existing file.

4. Specify the /T option to include additional trace information, such as header information about the command line options specified, full paths, and additional information on each table in the report.

Report Examples

The following command reports the referential constraints on the BTU named database. It redirects the output to the INFO.TXT file and appends it to that file. In addition, the referential constraints are sorted by parent table names.

```
riutil -list btu /o:sys:\ssql\demodata\info.txt /a /s:p
```

The following command reports the referential constraints on all named databases on the current server. It redirects the output to the INFO.TXT file.

```
riutil -list * /o:sys:\ssql\demodata\info.txt
```

CHECK

The CHECK command checks for orphan records and verifies the consistency of the files' referential integrity data (stored in each data file) and the dictionary's RI data (stored in the data dictionary files). CHECK can also log and delete orphan records, as well as repair inconsistent RI information.

Format

RIUTIL -CHECK database_name dependent_table /O:*vol:path* [option ...]

database_name	The name of the database for which to check referential integrity information.
dependent_table	The name of the dependent table to check. If you enter an asterisk (*) for this parameter, RIUTIL checks RI for all dependent tables in the specified named database. If the table name contains a space, you must enclose the table name in double quotes (").
/O: vol: path	The full path of the file you want RIUTIL to use in producing the CHECK report.

For *option* , you can specify any of the following options in any order:

/P: password	The master password of dictionary files associated with the specified named database. The SQL Interface requires a password if security is enabled for the database.

/A	Appends the generated list to an existing file, if you specify an existing file with the /O: option.
/T	Displays additional trace information in the output file. This option is helpful in diagnosing database problems.
/C	Checks for orphan records in the specified dependent table.
/W	Writes to an exception table up to 4 KB of each orphan record located in the specified dependent table. If you specify this option, you do not need to explicitly specify the /C option; it is implied.
/D	Deletes all orphan records from the dependent table. If you specify this option, you do not need to explicitly specify the /C option; it is implied.
/I	Checks for inconsistent file and dictionary RI data.
/R	Repairs inconsistent dependent table RI data in the data file, using the RI information stored in the data dictionary as the standard. If you specify this option, you do not need to explicitly specify the /I option; it is implied.
/?	Displays the syntax for the CHECK command.

If you do not specify a database name or dependent table, this command displays all available named databases, as in the following example.

```
Database Names on This Server

BTU

BTUDEMO
```

The command also displays a short usage message.

Checking for Orphan Records and Repairing Inconsistent RI Information

If you receive the message “Error opening table. Status = 73,” RIUTIL has detected inconsistencies in the RI information for the specified named database.

To check for orphan records and inconsistent RI data in all dependent tables of a database:

- Enter the following command sequence:

```
riutil -check [file] * /o:[path] /c /i
```

This command sequence does not affect the original database.

To check for orphan records, write them to an exception table, and delete them from the original table:

- Enter the following command sequence:

```
riutil -check btu * /o:sys\ssql\demodata\student.mkd /w /d /r
```

In addition, this command checks and repairs any inconsistent RI data it detects.

To repair the inconsistencies, follow these steps:

1. Use the Pervasive.SQL Setup utility (see "Modifying Named Databases") to disable the Integrity Enforced flag on the named database.

2. Run the CHECK command using the /R option to repair inconsistencies.

3. Use the Setup utility again to enable integrity enforcement on the named database.

Generating a List of Orphan Records or Inconsistencies

You use the /O: option to tell the utility how to direct the report information to an output file.

To generate a report that includes orphan records and/or inconsistencies:

1. Specify option /C and or option /I.The following report example output displays information on a named database with inconsistencies; it also repaired the RI data and checked for orphan records.

Example 10-2
Inconsistency and Orphan Record Verification Report

```
            **********************************************************************Foreign Key
                              STUDENTHASATUITION for Table STUDENT

                                         Parent Table Tuition
                                 Checking RI Data in Parent Table Tuition
                                              Dictionary Data    File Data
                    Version #                 : 7.00             7.00
                  Database Name              : BTU1               BTU2
                   Table Name                : Tuition          Tuition
                      Parent to Other Keys   : 1                 1
                     # Foreign Key Defs      : 0                 0

                                       RI Data is NOT consistent.
                                          RI Data Repaired.

                                        Beginning Orphan Check.
                  Total records in dependent table                  :1404
                   Total orphan records                               : 0
                    Total orphan records written to exception table   : 0
                   Total orphan records deleted from dependent table: 0
           End of Orphan Check.of check for Foreign Key: STUDENTHASATUITION
```

2. By default, the RI utility overwrites any existing information in the output file. Specify the /A option to append new information to the existing file.

3. Specify the /T option to include additional trace information, such as header information about the command line options specified, full paths, and additional information on each table in the report.

Generating Exception Tables

To generate an exception table and add it to the database dictionary:

- Specify the /W option.

  The exception table has the same location and file name as the original file, but with a .EXC extension. For example, if the RI utility generated an exception table on the Billing table in the BILLING.MKD file, the exception table would be named EXC_Billing and would be stored in a file named BILLING.EXC.

  In the following example, the CHECK /W command might generate the following output if the Billing table contained four orphan records. The orphan records would be written to an exception table, but they would not be deleted.

Example 10-3
Sample Exception Table

```
********************************************************Foreign Key
                    BILLINGBYREGISTRAR for Table BILLING

                               Parent Table Person
                      Checking RI Data in Parent Table Person
                               RI Data is consistent.

                            Beginning Orphan Check.
                  Exception table EXC_BILLING added to database.
                 Total records in dependent table                :1319
                  Total orphan records                             : 4
                   Total orphan records written to exception table   : 4
                  Total orphan records deleted from dependent table: 0
     End of Orphan Check.of check for Foreign Key: BILLINGBYREGISTRAR
```

The first column in the exception table record is an index column, and it contains the parent table name. The remainder of each record contains the same columns as each original orphan record, up to 4,090 bytes. The RI utility truncates records larger than 4,090 bytes in the exception table.

To generate a table containing the complete orphan records:

1. Execute a SQL statement that selects the orphan records from the original table and inserts them into another exception table, as in the following example:

```
INSERT INTO Orph_Billing* FROM BillingID = (SELECT ID FROM EXC_Billing);
```

2. Use a SQL Interface application such as SQLScope to view the contents of an exception table.

Converting MicroKernel Data Files

This chapter describes how to rebuild previous versions of MicroKernel files into version 7.0 format using the Rebuild utility. It also explains how to migrate Scalable SQL version 3.01 views to version 4.0 with the View Conversion utility.

For information about performing either of these operations, refer to the following sections:

- "Converting MicroKernel Data Files"
- "Migrating Scalable SQL 3.01 Views to 4.x"

Converting MicroKernel Data Files

Pervasive.SQL includes two versions of the Rebuild utility: an interactive version runs on Windows 95/Windows NT, Windows 3.x and OS/2, and a command-line version that runs as an NLM on NetWare. The Pervasive.SQL workstation engine only uses the Win32 version of the Rebuild utility.

The Rebuild utility can convert MicroKernel data files as shown in the following table.

Table 11-1
Rebuild Utility Conversions

Original File Format	Converted File Format	Reason for Conversion
pre-6.0	7.*x*	Take advantage of 7.*x* features and improve general performance.
6.*x*	7.*x*	Take advantage of 7.*x* features and improve general performance.
7.*x*	7.*x*	Original file does not have a system key.
pre-6.0	6.*x*	Take advantage of 6.*x* features and improve general performance. Use this option only if you are still running the 7.*x* engine with other 6.*x* engines.

The file format that results from the conversion depends on the value you set for the MicroKernel's Create File Version configuration option (see “Create File Version”), which you specify using the Setup utility. For example, if you set the Create File Version to 7.*x* and you run the Rebuild utility on 6.*x* files, the utility converts the files to 7.*x* format.

When you convert files to the 7.*x* format, the MicroKernel's System Data option (see “System Data”) controls whether the command-line Rebuild utility adds a system-defined log key in files that do not contain a unique key.

Before you run the Rebuild utility, back up all the data files you plan to convert. Having backup copies ensures against data loss if a power interruption occurs while you are running the utility. To ensure that your backup is successful, perform any one of the following operations:

- Close all data files before running the backup utility.
- Use continuous operations.
- Use a backup utility that opens the files in exclusive write mode so that other processes cannot write to the files. Ensure that the backup utility has exclusive rights to the files.

Note: You cannot run the Rebuild utility on a file that is in continuous operation mode.

The remainder of this section discusses the two versions of the utility:

- “Interactive Rebuild Utility”
- “Command-Line Rebuild Utility”

Interactive Rebuild Utility

This section provides instructions for using the interactive Rebuild utility which runs on Windows 3.x, Windows 95, Windows NT, and OS/2. The descriptions in this section are written from the Windows perspective, however, the OS/2 version is similar.

Running the Rebuild Utility

To run the Rebuild utility for Windows 3.x:

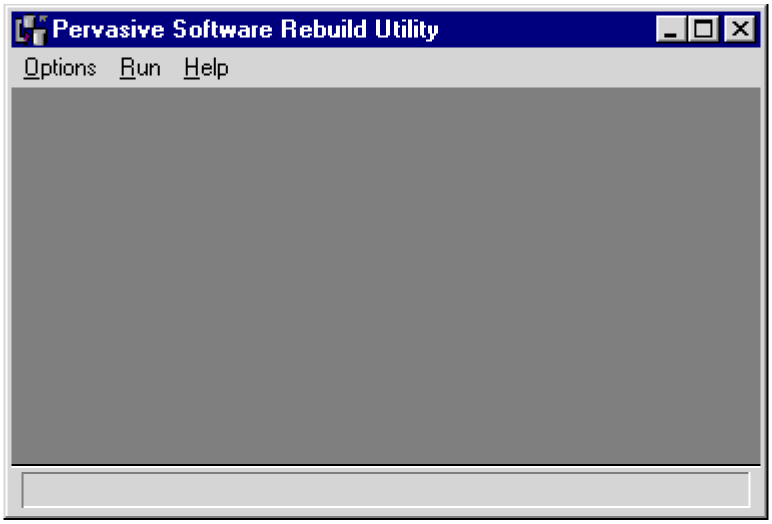
- In the **Pervasive.SQL 7** program group, double-click the Rebuild utility icon. The Rebuild utility's main window appears similar to Figure 11-1.

To run the Rebuild utility on Windows 95/Windows NT:

1. From the **Start** menu, select **Programs** and then **Pervasive SQL 7 Server** .

2. Click **Rebuild (Win32)** . The main window displays as illustrated in Figure 11-1.

Figure 11-1
Rebuild Utility Main Window



Getting Help

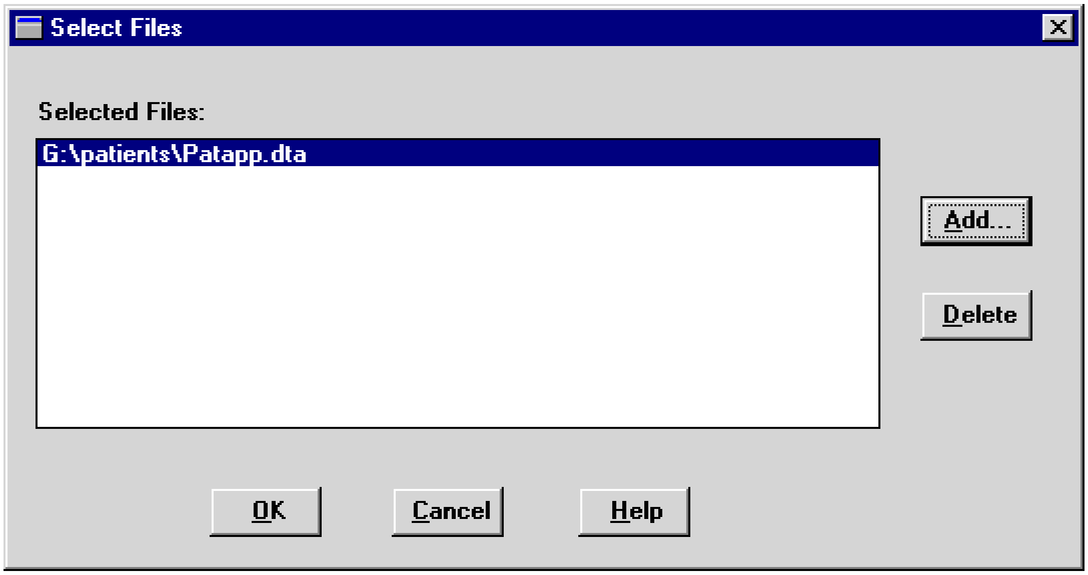
You can access a Help file from either the **Help** menu or by clicking **Help** in any dialog box.

Converting a Data File

To convert a data file:

1. Choose **Select Files** from the **Options** menu. The **Select Files** dialog box appears (Figure 11-2.)

Figure 11-2
Select Files Dialog



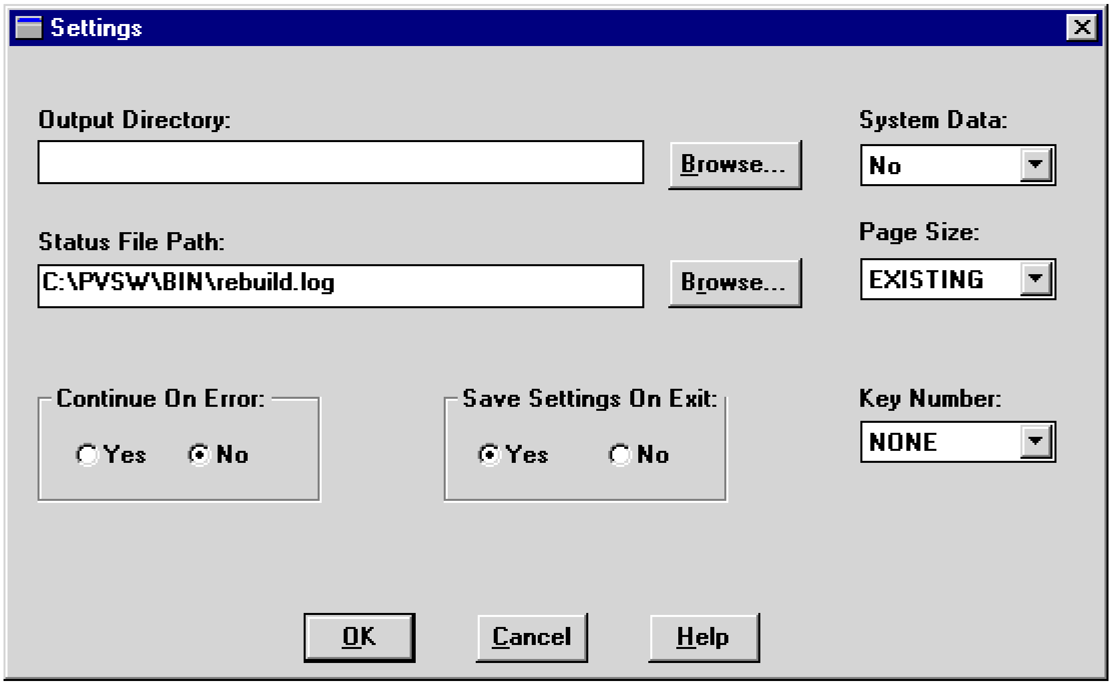
G:\patients\Patapp.dta

2. Click **Add** and select the file you want to rebuild. You can select more than one file to rebuild at a time. Click **OK** when you have finished adding files to rebuild.

   The Rebuild utility deletes the original file after rebuilding it if the file is being rebuilt in the same directory. If the new file is in a different directory, the original file is not deleted.

3. Before you rebuild the file(s), you may want to specify settings. Choose **Settings** from the **Options** menu. The Settings dialog displays as illustrated in Figure 11-3.

Figure 11-3
Settings Dialog



Settings

Output Directory:

Status File Path: C:\PVSW\BIN\rebuild.log

Continue On Error: ○ Yes ● No

Save Settings On Exit: ● Yes ○ No

System Data: No

Page Size: EXISTING

Key Number: NONE

You can change the configuration options for the Rebuild utility before you rebuild your selected file or files. These options are defined in Table 11-2.

Table 11-2
Controls in the Settings Dialog

Control	Description
Output Directory	Specifies an alternate location for the rebuilt files. (The default location is the current directory.) You must specify a directory that already exists. This option lets you rebuild large files on a different server. The MicroKernel and its communications components must be loaded on the server that contains the rebuilt files. Do *not* use wildcard characters in the path. If the Output Directory location is different than the original file’s location, the original file is not deleted during the rebuild. If the output directory is the same as the original file, the original file is deleted upon completion of the rebuild.
Status File Path	Specifies a location for the rebuild log file. (The default location is the current working directory.) Do *not* use wildcard characters in the path.
Continue on Error	Determines whether the Rebuild utility continues if it encounters an error during the rebuild process. If you select Yes, the utility continues with the next file even if an error occurs. The utility notifies you of non-MicroKernel data files or other errors but continues rebuilding data files. If you select No, the utility halts the rebuild if it encounters an error. This option is useful if you have specified wildcard characters for the rebuilt files.
Save Settings Upon Exit	Saves the current values in this dialog for use in subsequent Rebuild sessions.
System Data	Specifies whether files are rebuilt with system data. The MicroKernel cannot

perform logging for a file with no system-defined log key when no user-defined unique key exists.

Page Size

Specifies the page size (in bytes) of the new files. Choose either EXISTING, 512, 1024, 2048, 3072, or 4096. If you select EXISTING, the utility uses the existing page size. The utility changes the page size if the original size does not work.

For example, assume you have a v5.*x* file with a page size of 1,024 and 24 keys. Because Btrieve 6.0 and later supports only 23 keys for a page size of 1,024, the utility automatically selects a new page size for the file and writes an informative message to the status file.

Key Number

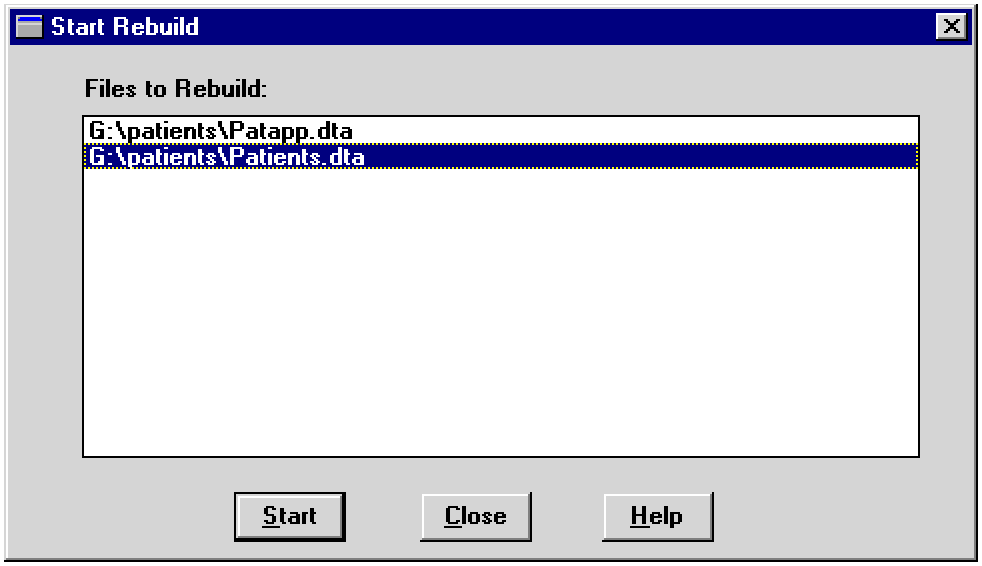
Specifies the key by which the utility reads when rebuilding a file. If you specify NONE for this option, the utility clones the files, drops the indexes, copies the records into the new files, and rebuilds the indexes. Because this method is faster and creates smaller files than specifying a key number, use it whenever possible.

This method may create a new file in which the records are in a different physical order than in the original file.

If you specify a key number, the utility clones and copies the files without dropping and replacing indexes. While this method is slower than specifying NONE, it is available in case you do not want to rebuild your indexes.

4. After you specify the settings, you need to start the file conversion process. Select **Start Rebuild** from the **Run** menu. The Start Rebuild dialog displays as indicated in Figure 11-4.

Figure 11-4
Start Rebuild Dialog



Start Rebuild

Files to Rebuild:

G:\patients\Patapp.dta
G:\patients\Patients.dta

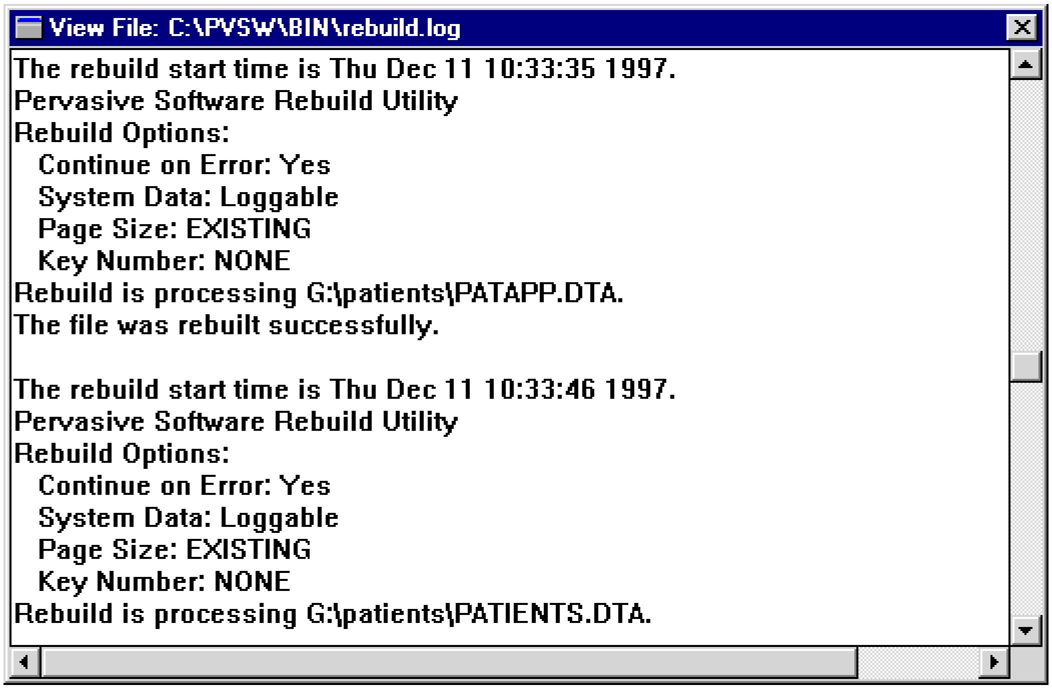
5. Select the file you want to convert and then click **Start** to begin the rebuild process.

When the process completes, a message dialog informs of the success or failure of the conversion and prompts you to view the results.

6. Click **Close** when you have finished converting files.

7. To display the results, select **View Status File** from the **Run** menu. The REBUILD.LOG file is displayed as illustrated in Figure 11-5.

Figure 11-5
Start Rebuild Dialog



```
The rebuild start time is Thu Dec 11 10:33:35 1997.
Pervasive Software Rebuild Utility
Rebuild Options:
  Continue on Error: Yes
  System Data: Loggable
  Page Size: EXISTING
  Key Number: NONE
Rebuild is processing G:\patients\PATAPP.DTA.
The file was rebuilt successfully.

The rebuild start time is Thu Dec 11 10:33:46 1997.
Pervasive Software Rebuild Utility
Rebuild Options:
  Continue on Error: Yes
  System Data: Loggable
  Page Size: EXISTING
  Key Number: NONE
Rebuild is processing G:\patients\PATIENTS.DTA.
```

The Rebuild utility writes to the status file for every file it attempts to convert. The log file (REBUILD.LOG by default) is an ASCII text file that is placed by default in the directory in which you run the Rebuild utility from.

You can examine the log file by selecting the View Status File command from the Run menu. The rebuild settings are listed for every file. If you disabled the **Continue on Error** setting, the status file contains the information up to the point of the error. If the rebuild was not successful, the status file contains error messages explaining why the rebuild failed.

Command-Line Rebuild Utility

This section provides detailed instructions for using the command-line Rebuild utility which runs only as an NLM on NetWare.

Running the Rebuild Utility on Netware

To run the Rebuild utility for NetWare:

1. Run RCONSOLE from a workstation, or go to the server’s console.

2. Enter one of the following commands at the prompt:

LOAD BREBUILD [–*option* ...] *file*

or

LOAD BREBUILD *@command_file*

Changing Configuration Options

To change the configuration options for the Rebuild utility for NetWare:

The *Option command* specifies the configuration options for the utility. Precede each option letter with a dash (–). Do not place a space between the dash and the option letter or between the option letter and its value. You can enter the option letter in either uppercase or lowercase.

–B[*path*] Specifies an alternate location for the rebuilt files. (The default location is the current directory.) You can also specify a different server with this option. On your local server, you must have the MicroKernel Database Engine and the Message Router loaded. On a remote server, you need the MicroKernel Database Engine and communications components loaded. Do *not* use wildcard characters in the path you specify.

–C Instructs the utility to continue with the next file even if an error occurs. The utility notifies you of non-MicroKernel data files or other errors but continues rebuilding data files.

This option is useful if you have specified wildcard characters for the rebuilt files.

–D Converts pre-6.0 supplemental indexes (which allow duplicates) to 6.*x* or 7.*x* indexes with linked-duplicatable keys. (By default, the utility preserves the indexes as repeating-duplicatable keys.) If you access your data files only through Btrieve and your files have a relatively large number of duplicate keys, you can use this option to enhance the performance of the Get Next and Get Previous operations.

If you are using Scalable SQL to access your data files, do *not* use the –D option.

–M0 | –M2 Specifies the conversion method, as follows:

–M0 Clones and copies the files without dropping and replacing indexes. While this method is slower than M2, it is available in case you do not want to rebuild your indexes.

–M2 (Default) Clones the files, drops the indexes, copies the records into the new files, and rebuilds the indexes. Since this method is faster and creates smaller files than the M0 method, use it whenever possible.

The M2 method may create a new file in which the records are in a different physical order than in the original file.

–P[*nnn*] Specifies the page size (in bytes) of the new files. If you specify –P with no page size, the utility chooses the optimum page size for your file.

If you do not specify the –P parameter, the utility changes the page size if the original size does not work.

For example, assume you have a v5.*x* file with a page size of 1024 and 24 keys. Since Btrieve 6.0 and later supports only 23 keys for a page size of 1024, the utility automatically selects a new page size for the file and displays an informative message on the screen.

–K[*nn*] Specifies the key by which the utility reads when rebuilding a file. If you do not specify this option, the utility reads the file in physical order.

–T Does not preserve the Transaction Tracking System (TTS) bit during conversion. If you specify this option, the utility clears the TTS bit if it was set. If you do not specify this option, the utility sets the TTS bit when creating the new file if the original file had the TTS bit set.

File and *@command_file* are defined as follows:

file Specifies the set of files to convert. Use full directory names, including the volume name. You may use wildcards (* and ?) in these file names. The Rebuild utility applies the original file's owner name and level to the new file.

@command_file Specifies a command file for the utility to execute. You can include multiple entries in one command file. Each entry in the command file contains the utility options (if any) and the set of files to convert, followed by <end> or [end]. When specifying the files to convert, be sure to use full directory names, including the volume name. You can use wildcards (* and ?) in these file names.

The following is an example of a Rebuild utility command file:

```
–C sys:\mydir\*.* <end>
–C –P1024 dta:\dir\*.* <end>
–M0 –K0 sys:\ssql\*.* <end>
```

The following example places the rebuilt files in a different directory on the server:

```
LOAD BREBUILD –Bsys:\newfiles –C –P4096 sys:\oldfiles\*.mkd
```

Viewing the BREBUILD.LOG File

After rebuilding your files, check the utility's log file to see if any errors occurred during the conversion. The BREBUILD.LOG is an ASCII text file, which is placed in the SYS:\SYSTEM directory. You can examine the log file using a text editor.

Deleting Temporary Files

By default, the Rebuild utility creates temporary files in the same directory in which the conversion takes place. Therefore, you need enough disk space in that directory (while the Rebuild utility is running) to accommodate both the original file and the new file. You can specify a different directory for storing these files using the Output Directory option in the interactive version of the utility or using /B option in the NetWare version.

Normally, the Rebuild utility deletes temporary files when the conversion is complete. However, if a power failure or other serious interruption occurs, the Rebuild utility may not delete the temporary files. If this occurs, look for file names such as _T-*xxxxx* .TMP and delete them.

Migrating Scalable SQL 3.01 Views to 4.x

The View Conversion utilities are a pair of Win32 programs running on Win32 platforms that enable existing Scalable SQL users to display Scalable SQL 3.01 views and migrate those views to Scalable SQL 4.x. This utility resides in your c:\pvsw\bin directory and only is installed if you choose the SQL Client installation. The following subsections describe the various elements of the utilities:

- "Special Notes on the View Conversion Utilities"
- "Starting the View Conversion Utilities"
- "Converting Stored View Definitions to Text"

Special Notes on the View Conversion Utilities

You must convert all existing views in order to use them with Scalable SQL 4.x. The utilities are as follows:

VTEXT32.EXE	Displays the definition for a stored view.
VCONV32.EXE	Converts a stored view definition into a view you can use in a Scalable SQL v4.0 database.

The SQL Engine creates views for the API level at which you created them in Scalable SQL v3.01. Therefore, if you issue a CREATE VIEW statement and attempt to recall the view using an application that uses the relational primitives, you receive an error. However, if the View Conversion utility stored the view as a relational primitive view, you can use the view name in a SQL statement.

Generating a List of Views in a Dictionary

The View Conversion utilities operate on a single view at a time. For a list of all views defined in your dictionary, execute the following statement using SQLScope:

```
SELECT DISTINCT Xv$Name FROM X$View
```

Using the View Conversion Utilities on Windows NT

Windows NT strips out special characters, such as the caret (^) when it processes the command line request. Therefore, if you run the View Conversion utility from an NT DOS session and the view names contain special characters, such as ^, you need to enclose the view name in double quotes when you run the VTEXT32 or VCONV32 utility. Also, enclose the command-line setting for the blank replacement character, -B in double quotes. For example, if your blank replacement character is the caret (^), enter "-B^".

Starting the View Conversion Utilities

ℹ

For NetWare:

1. Enter either:

```
VTEXT32 [options] <view name>
```

or

```
VCONV32 [options] <view name>
```

For Windows NT:

1. Enter either:

```
VTEXT32 [options] <view name>
```

or

```
VCONV32 [options] <view name>
```

> **Note:** If you are using Pervasive.SQL for Windows NT, you must run this utility from a workstation connected to Windows NT.

Converting Stored View Definitions to Text

The VTEXT32 utility converts a stored view definition to a text string and displays the string. Optionally, you can store the string in a file. You can display only one view at a time. VTEXT32 works only with pre-v4.0 Scalable SQL databases.

The syntax for VTEXT32 is as follows:

```
VTEXT32 [options] <view name>
```

<view name> is the name of a view in a Scalable SQL database.

The following table lists valid options you can use with VTEXT32:

Table 11-3
VTEXT32 Utility Options

Option	Description	
-<type>	Where <type> can be one of the following:	
	0	SQL views return the CREATE VIEW statement, and primitive stored views return only primitive-level information. This is the default option.
	1	Generate SQL text.
	2	Generates information about the primitive-level stored view.
	3	Returns stored view version number.
	4	Generates information about columns in the view.
-B<c>	<c> is the character to use for blank replacement. The default is a caret (^).	
-O<name>	<name> is the name of the output file.	
-D<name>	<name> is the directory containing	

	the dictionary files. The default is the current directory.
-P<name>	Where <name> is the master user's password.
-V<name>	<name> is the directory containing VIEW.DDF. The default is the current directory.

Example

For a database that contains a view called "COMP_ADDR_ORPHANS," the following command produces output at your console in the form of a valid SQL statement that defines the view.

```
VTEXT32 -1 -UMYNAME -PMYPASS COMP_ADDR_ORPHANS
```

Result:

```
CREATE VIEW COMP_ADDR_ORPHANS AS SELECT Company_Id,Address_IdCompany ANOT EXISTS (SELECT *
FROM ADDRESS B WHERE A.ADDRESS_ID =          B.ADDRESS_ID)
```

The following command stores the same output in the file VTEXT32.OUT instead of displaying it at your console.

```
VTEXT32 -1 -UMYNAME -PMYPASS -OVTEXT32.OUT COMP_ADDR_ORPHANS
```

Converting Stored View Definitions to Scalable SQL 4.x

The VCONV32 utility converts a stored view definition to a Scalable SQL 4.x view. VCONV32 works only with pre-4.0 Scalable SQL databases.

The primary purpose of VCONV32 is to convert the views in a pre-4.x Scalable SQL database to Scalable SQL 4.x format. Scalable SQL 4.x and Btrieve must be running on the Windows NT or NetWare file server in order to run VCONV32.

Following is the syntax for VCONV32:

```
VCONV32 [options] <viewname>
```

The <viewname> is the name of a view in a Scalable SQL database.

The following table displays a list of the valid options you can use with VCONV32:

Table 11-4
VCONV32 Utility Options

Option	Description
-<type>	Where <type> can be one of the following: 0: Creates view in the format of the existing view definition. 1: Creates SQL format view. 2: Generates primitive format view.

	5: Generates primitive format view; however, if the view requires SQL functionality, it stores it in SQL format. This is the default option
-B<c>	<c> is the character to use for blank replacement.
-D<name>	<name> is the directory containing the dictionary files. The default is the current directory.
-F<paths>	Where paths are the directories to locate table data files with paths separated by ';'.
-N<name>	Where <name> is the new view name. The default is the original view name.
-P<name>	Where <name> is the Master user's password.
-V<name>	<name> is the directory containing VIEW.DDF. The default is the current directory.

Running VCONV32

The following steps describe a typical procedure for running VCONV32:

1. Back up your dictionary files.

2. Copy the VIEW.DDF file to another directory and delete the copy of VIEW.DDF in the directory with the rest of your DDFs.

> **Note:** During operation, the view conversion utility, VCONV32.EXE, creates a new copy of VIEW.DDF in Scalable SQL v4.0 format in the directory in which the other DDF files are located.

3. Enter the following command (all on one line) to execute the view conversion process:

```
VCONV32 -V<directory location of old VIEW.DDF>
D<directory location of other DDF files>
P<password>
<VIEWNAME>
```

You can convert only one view at a time. If you have several views to convert, create a batch file that invokes VCONV32 on each view defined in the database, and redirect the output to a text file. Doing so allows you to review the results later to determine if the view conversion process was successful.

For example, if you were in a directory containing the DDF and data files with views you wanted to convert, and you copied your VIEW.DDF file to a subdirectory called SAVEVIEW, and you had a batch file called DOVIEWS.BAT containing these lines:

```
VCONV32 -VSAVEVIEW ADDRESS_ORPHANS-VSAVEVIEW COMP_ADDR_ORPHANS-VSAVEVIEW
COMP_ADD_ORPHANS-VSAVEVIEW CONTACT_ORPHANS-VSAVEVIEW CONT_ADD_ORPHANS
```

Executing the following batch file would convert the views ADDRESS_ORPHANS, COMP_ADDR_ORPHANS, COMP_ADD_ORPHANS, CONTACT_ORPHANS, and CONT_ADD_ORPHANS to Scalable SQL 4.0 compatible views, saving the output from the conversion process in the file DOVIEWS.OUT.

```
DOVIEWS >DOVIEWS.OUT
```

If the VCONV32 utility returns an error that it is unable to convert a view, use the VTEXT32 utility (option 0 or 1) to display the view definition. This may help determine the cause of the error in the VCONV32 utility. If you are unable to resolve the problem, please contact Pervasive Customer Support for assistance in converting your views.

Smart Component Type Codes

This appendix displays the type codes for Pervasive.SQL components.

Component Type Codes Table

Table A-1
Component Type Codes

Component Group	Component	Component Type
Abstract OS Services	Interface DLL	AIF
	Glue DLL	SCM
Btrieve	Interface DLL	BIF
	Glue DLL	BTR
Communications	Client Requester	NSL
	Requester Win95 Support	NSR
	BSPXCOM.NLM	BSP
	SSPXCOM.NLM	SSP
	BTCPCOM.NLM	BIP
	STCPCOM.NLM	SIP
	NWBSRVCM.NLM	BSV
	NWSSRVCM.NLM	SSV
Database Names	Interface DLL	DIF
	Glue DLL	DBN
Install Scout	Resource DLL (English)	IRE
	API Test and Analysis	ATA
	Communication Diagnosis and Analysis	CDA
MKDE	Interface DLL	MIF
	Server Engine	MSE
	Server Engine Resource File	MSR
	Local System File	MLC
Scalable SQL	Interface DLL	SIF
	Glue DLL	SQL
	Server Engine	SSE
	Workstation Engine	SCE
	Scalable SQL Stub (e.g. NTSSQL.EXE)	SST
	Scalable SQL Convert/Mask DLL	SFM
	Scalable SQL Local Engine Access Module	SLC
	Scalable SQL 16-32 Thunking DLL	STK
User Count	Manager	UCM

Manager Resource File	UCR
Sys File	UCS
User Interface	UUI
Resource	URC
SQLScope	SCP
SQLScope BG Export	SPE
SQLScope BG Import	SPI
SQLScope BG Run All	SPA
SQLScope BG RI Check	SPC
SQLScope BG RI List	SPL
SQLScope SQLUTIL DLL	SPD
SQLScope SQLUTIL Resource	SPR
RIUTIL (Referential Integrity)	RMC
SQLUTIL (Scalable SQL Maintenance)	SMC
SQLUtil Resource	SMR
DDF Ease	DDF
DDF Ease Resource DLL	DDR
SmartScout	SSC
InstallScout	ISC
Utilities Services DLL (BTISC.DLL)	SVD
Utilities Services Executable (BTISC.EXE)	SVC
User Count Initialization	UCI
User Count Initialization Resource DLL	UIR
User Count Administrator	UCA
User Count Administrator Resource DLL	UAR
Rebuild	RBD
Rebuild Resource DLL	RBR
Btrieve Interactive Maintenance	BMG
Btrieve Interactive Maintenance Resource DLL	BGR
BUTIL (Command-Line Maintenance)	BMC
BUTIL Resource DLL	BCR
Btrieve Function Executor	FEX
Btrieve Function Executor Resource DLL	FER
Monitor	MON

	Monitor Resource DLL	MOR
	Setup	SET
	Setup Resource DLL	SER
Utilities Requester	Interface DLL	UPI
Misc. Client Components	Client Resource Strings	CRS
	Splash Screen EXE	LGO

Description Files

A *description file* is an ASCII text file that contains descriptions of file and key specifications that the Maintenance utility can use to create data files and indexes. Some users employ description files as a vehicle for archiving information about the data files they have created. Description files are not the same as DDFs, or Data Dictionary Files, which are used with the SQL Interface and the ODBC Interface.

Description files contain one or more elements. An element consists of a keyword, followed by an equal sign (=), followed by a value (with no space). Each element in a description file corresponds to a particular characteristic of a data file or key specification.

> **Note:** Before using description files, you should be familiar with Btrieve fundamentals, such as data compression and index balancing. For information about these topics, refer to the *Pervasive.SQL* Programmer's Guide.

This appendix discusses the following topics:

- "Rules for Description Files"
- "Description File Examples"
- "Description File Elements"
-
-
-

Rules for Description Files

Use the following rules when creating a description file.

- Enter elements in either uppercase or lowercase.
- Separate elements from each other with a separator (blank space, tab, or carriage return/line feed), as in the following example:

```
record=4000

key=24
```

- Specify the description file elements in the proper order. Table B-1 presents the elements in the appropriate order.
- Address all element dependencies. For example, if you specify nullkey=allsegs in your description file, you must also specify a value for the value= element.
- Define as many keys as you specify with the Key Count element. For example, if you specify key=12 , you must define 12 keys in the description file.
- For a key that consists of multiple segments, you must define the following elements for each key segment:
    - Key Position
    - Key Length
    - Duplicate Key Values
    - Modifiable Key Values
    - Key Type

  The Descending Sort Order element is optional for each segment.
- If any key in the file uses an ACS, you must specify an ACS file name, a country ID and code page ID, or an ISR table name. You can include this information as either the last element of the key (applies to current key only) or the last element in the description file (applies to entire data file).
    - You can specify only one ACS per key, and you must provide an ACS file name, country ID and code page ID, or an ISR table name. Different keys in the same file can use different types of ACSs; for example, Key 0 can use an ACS file name, and Key 1 can use a country ID and code page ID.
    - Different segments of the same key cannot have different ACSs.
    - If you specify an ACS at the end of a description file, it is used as the default ACS. That is, if you specify alternate=y for a given key but do not include an ACS file name, country ID and code page ID, or an ISR table name for that key, the MicroKernel uses the ACS file name, country ID and code page ID, or ISR table name specified at the end of the file.
    - If you are creating a new key and you specify alternate=y but you omit the ACS file name, country ID and code page ID, or ISR table name, the MicroKernel does not create the key.
- If a description file element is optional, you can omit it.
- Make sure the description file contains no text formatting characters. Some word processors embed formatting characters in a text file.

Description File Examples

The sample description files shown in this section describe a data file. This data file has a page size of 512 bytes and 2 keys. The fixed-length portion of the record is 98 bytes long. The file allows variable-length records but does not use blank truncation.

The file uses data compression, allows for Variable-tail Allocation Tables (VATs), and has the free space threshold set to 20 percent. The MicroKernel Database Engine preallocates 100 pages, or 51,200 bytes, when it creates the file. The file has two keys: Key 0 and Key 1. Key 0 is a segmented key with two segments.

In Figure B-1, both keys use the same ACS file name (UPPER.ALT). In Figure B-2, both keys use the same country ID (–1) and code page ID (–1). In Figure B-3, Key 0 and Key 1 use different ACS file names (LOWER.ALT and UPPER.ALT, respectively). In Figure B-4, the file has no keys except the system-defined key used for logging.

Figure B-1
Sample Description File Using Alternate Collating Sequence File Name

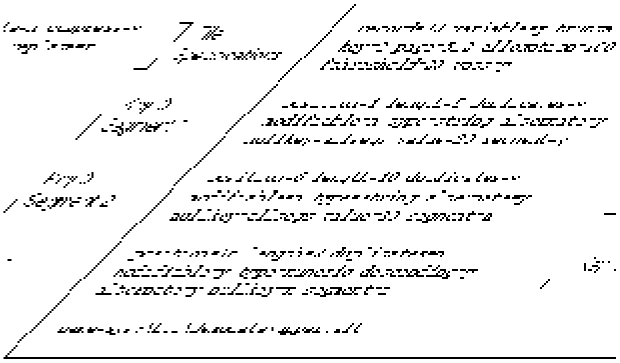


Figure B-2
Sample Description File Using Alternate Collating Sequence ID

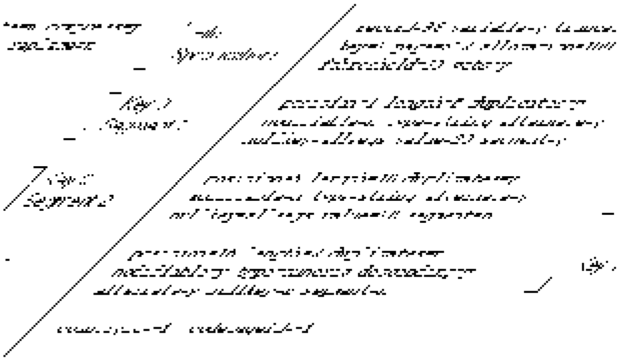
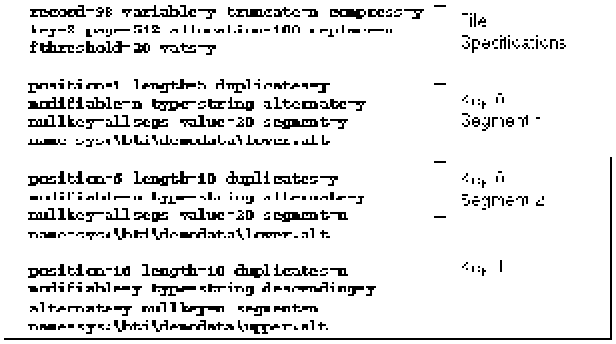
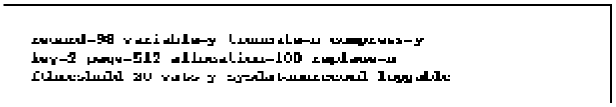


Figure B-3
Sample Description File Using Alternate Collating Sequence File Name on a Key Segment



```
record=98 variable=y truncate=n compress=y
key=2 page=512 allocation=100 replace=n
fthreshold=20 vats=y

position=1 length=5 duplicates=y
modifiable=n type=string alternate=y
nullkey=allsegs value=20 segment=y
name=sys:\btr\demodata\lower.alt

position=6 length=10 duplicates=y
modifiable=n type=string alternate=y
nullkey=allsegs value=20 segment=n
name=sys:\btr\demodata\lower.alt

position=16 length=10 duplicates=n
modifiable=y type=string descending=y
alternate=y nullkey=n segment=n
name=sys:\btr\demodata\upper.alt
```

Figure B-4
Sample Description File Using System-Defined Key for Logging



```
record=98 variable=y truncate=n compress=y
key=2 page=512 allocation=100 replace=n
fthreshold=20 vats=y sysdataonrecord=loggable
```

Description File Elements

Description file elements must appear in a particular order. Table B-1 lists the description file elements in the appropriate order. For each element, the table specifies the required format and the range of acceptable values.

- An asterisk (*) indicates that the element is optional.
- A pound sign (#) indicates that it is not applicable in the current MicroKernel version but is retained for backward compatibility with previous MicroKernel versions.
- A percent sign (%) indicates that the element is applicable only to the current MicroKernel version.

Table B-1
Summary of Description File Elements

Element	Keyword and Format	Range	Comments
File Specification Information			
Comment Block*	/*. . . . . */	5,120 bytes	None.
Record Length	record=*nnnn*	4–4,088	None.
Variable-Length Records	variable=<y|n>	N/A	Not applicable to key-only files.
Reserved Duplicate Pointer*	dupkey=<*nnn* >	0–119	Applicable only to files for which you plan to add linked-duplicatable keys.
Blank Truncation*	truncate=<y|n>	N/A	Not applicable for files that use data compression.
Data Compression*	compress=<y|n>	N/A	Not applicable to key-only files.
Key Count	key=*nnn*	0–119	Specify 0 to create a data-only file.
Page Size	page=*nnnn*	512–4,096	Must be a multiple of 512.
Page Preallocation*	allocation=*nnnnn*	1–65,535	None.
Replace Existing File*#	replace=<y|n>	N/A	None.
Include Data*	data=<y|n>	N/A	Specify *n* to create a key-only file. Cannot create a key-only file that both allows duplicates and uses a system-defined key.
Free Space Threshold*	fthreshold=<5|10|20|30>	N/A	Applicable only for files that have variable-length records. The default is 5.

Variable-Tail Allocation Tables (VATs)	huge=<y|n> #
	vats=<y|n>
		N/A	Applicable only for files that have variable-length records.

Balanced Index*	balance=<y|n>	N/A	None.

Use Key Number *	usekeynum=<y|n>	N/A	Used with the Key Number element.
**1Use System Data*%	sysdataonrecord=<n|loggable>	N/A	If no element specified, MicroKernel configuration is used. If creating a key-only file, MicroKernel configuration is used and this element is ignored. Also, you cannot create a key-only file that both allows duplicates and uses a system-defined key.
Key Specification Information			
Key Number *	keynum=*nnnn*	0–118	Must be unique to the file, specified in ascending order, and valid for the file's Page Size. Applicable only when creating a file.
Key Position	position=*nnnn*	1–4,088	Cannot exceed the Record Length.
Key Length	length=*nnn*	key type limit	Cannot exceed the limit dictated by the Key Type. For binary keys, the key length must be an even number. The total of the Key Position and Key Length cannot exceed the file's Record Length.
Duplicate Key Values	duplicates=<y|n>	N/A	Cannot create a key-only file that allows duplicates and uses a system-defined key.
Modifiable Key Values	modifiable=<y|n>	N/A	None.
Key Type	type=*validMKDEKeyType*	N/A	Can enter the entire word (as in float) or just the first three letters (as in flo).
Descending Sort Order*	descending=<y|n>	N/A	None.
Alternate Collating Sequence	alternate=<y|n>	N/A	Applicable only for case sensitive STRING, LSTRING, or ZSTRING keys. When creating an additional index for an existing file, if you want the index to use an ACS other than the first one in the data file, use with caseinsensitive=y .
Case-Insensitive Key*	caseinsensitive=<y|n>	N/A	Applicable only for STRING, LSTRING, or ZSTRING keys that do not use an ACS.
Repeating Duplicates*	repeatdup=<y|n>	N/A	If creating a key-only file, use repeating duplicates. If using this element, you must use duplicates=y.

Null Segments*	nullkey=<allsegs | n | anyseg |>	N/A	None.
Null Key Value	value=*nn*	1-byte hex	Used with the Null Segments element.
Segmented Key	segment=<y|n>	N/A	None.
Alternate Collating Sequence File Name/ID	name=*sequenceFile* or countryid=*nnn* and codepageid=*nnn* *isr=table name (%)*	valid path *or* values valid to operating system or –1	Used with the Alternate Collating Sequence element.

***1* When the MicroKernel adds a system key, the resulting records may be too large to fit in the file’s existing page size. In such cases, the MicroKernel automatically increases the file’s page size to the next accommodating size.

Glossary

This glossary contains terms referenced throughout the Pervasive.SQL product documentation set.

Pervasive.SQL Terms

Accelerated

In Btrieve 7.x, a file open mode that provides improved response time over Normal mode when updating data files. However, Accelerated mode disables the MicroKernel's logging capability. Therefore, the MicroKernel cannot guarantee transaction durability or atomicity on files opened in Accelerated mode.

If you are using Btrieve 6.x, Accelerated mode is equivalent to Normal mode, except that opening a data file in Accelerated mode cancels the effect of flagging a file as transactional.

If you are using a Btrieve 6.x workstation engine, the MicroKernel opens the file in Single Engine File Sharing (SEFS) mode (regardless of any other file sharing modes specified either implicitly or explicitly). A file opened in Accelerated mode allows only one workstation MicroKernel (and therefore only the tasks associated with that engine) to access the file. Also, the 6.*x* workstation MicroKernel ignores the NetWare Transaction Tracking System (TTS) flag on a file.

In pre-6.0 Btrieve, this file open mode disabled the engine's data recovery capability. Also, once a file was opened in Accelerated mode, other tasks could open the file only in Accelerated mode.

See also Exclusive, file open mode, file sharing, Normal, Read-Only, and Verify.

access modules

Implementations of specific data models that provide appropriate data structures and access techniques. Access modules receive requests from application programs and make calls to the MicroKernel to perform the required core data operations.

access path

See index path.

access right

A security right that determines a user's ability to access tables or individual columns. The access rights are Select, Update, Insert, Delete, Alter, References, and All.

See also All right, Alter right, Delete right, Insert right, References right, Security right, Select right, and Update right.

aggregate column value

A column value which is determined by a group aggregate function. This value is based on a set of column values selected from a table. *See also* group aggregate function.

aggregate function

One of a group of SQL functions you use to return a single result for a given set of column values. The aggregate functions are AVG, COUNT, MIN, MAX, and SUM. You can use aggregate functions in a select list or a HAVING clause.

aggregate value restriction clause

A type of restriction clause that follows the HAVING keyword in which the first expression of each condition must be a group aggregate function. The second expression can be a constant, a substitution variable, a string or numeric literal, or a subquery. An aggregate value restriction clause can contain multiple conditions. *See also* group aggregate function *and* restriction clause.

alias

A temporary name you can assign to a table or view in any of the following:

- The FROM clause of a SELECT or DELETE statement
- The INTO clause of an INSERT statement
- The table list in an UPDATE statement

Once you define an alias for a particular table or view, you can use the alias elsewhere in the same statement to qualify column names in that table or view.

All right

A Scalable SQL security right that includes the Select, Insert, Update, Delete, Alter, References, and Create Table rights.

all-segment null

A key attribute that instructs the MicroKernel to exclude a particular record from the index only if the value of all key segments of that record matches the specified null value. In pre-6.0 Btrieve, all-segment null keys are known as manual keys. *See also* null key and any-segment null.

Alter right

A Scalable SQL security right that allows users or user groups to modify table definitions. *See also* table definition.

alternate collating sequence

A sorting sequence that the MicroKernel uses to sort string values differently from the way that the standard ASCII collating sequence sorts them. Alternate collating sequences (ACSs) can be locale-specific, user-defined, or an international sort rule (ISR). *See also* collation table and International Sorting Rules (ISR).

alternate key

In Scalable SQL, a candidate key that is not a primary key. It also uniquely identifies a row in a table.

anomaly

An illegal RI definition that, if permitted, could result in the inconsistent handling of certain update, insert, or delete operations, thus harming the integrity of the data.

any-segment null

A key attribute that instructs the MicroKernel to exclude a particular record from the index if the value of any key segment of that record matches the specified null value. In pre-6.0 versions of Btrieve, any-segment null keys are known simply as null keys. *See also* null key and all-segment null.

API

See Application Programming Interface (API).

application

A program or set of programs, such as a spreadsheet or a payroll application, that performs a task or a group of related tasks. Also, a program written by or for users to assist them in their work. *See also* task.

application interface

A particular programming language interface (such as C or Pascal) that allows access to data files from an application.

Application Programming Interface (API)

A set of functions that an application uses to access a database or initiate system-level routines. A particular programming language interface (such as C or Pascal) to an API. A program that allows access to MicroKernel files from an application program.

archival logging

A MicroKernel capability that, when activated, records all the operations that change a specified data file. These changes are recorded in an archival log file. In the event of a system failure, you can configure the MicroKernel to use the archival log file to *roll forward* (recover) changes made to the data file between the time archival logging was initiated and the time of the system failure.

arithmetic operator

An expression operator you can apply to two expressions that represent numeric data (either numeric constants or numeric columns). The arithmetic operators are addition (+), subtraction (–), multiplication (*), division (/), integer division (//), and MOD (%).

ascending

In Scalable SQL, the default collating order for an index. In Btrieve, the default collating order for a key. *See also* sort order.

ASCII

An acronym for American Standard Code for Information Interchange. ASCII is an 8bit information code with which computers can translate letters, numbers, control codes, and punctuation into digital form.

attribute

See column attribute and index attribute.

back end

A program that provides computationally intensive processing and relies on a front end to provide a user interface. For example, a back end may specialize in calculation, communication services, or data management. Back ends are also called *engines* . *See also* database server and front end.

base column

A column containing data from a base table.

base table

A table directly associated with a physical data file that contains the actual rows and columns of the table. Unlike a virtual table, whose data may or may not be directly associated with a physical table. *See also* data dictionary and table**.**

blank fill

To add a series of blanks to the end of a character string to make it a desired number of characters in length.

blank replacement character

A character you must use to replace blanks in a dictionary name when you pass the name to Scalable SQL. Otherwise, Scalable SQL cannot distinguish between the blanks in dictionary names and the blanks between elements in an expression. Valid blank replacement characters are the caret (^), underscore (_), and tilde (~). Do not use one of these characters when defining a name if you intend to use that character as the blank replacement character.

blank truncation

A method for conserving disk space by not storing the trailing blanks in the variable-length portions of records when the records are written to the file.

Boolean

A category of data types you can use to represent true/false logical values, or any two-valued data. The Boolean data types include BIT and LOGICAL.

Boolean operator

An operator that specifies a logical condition: NOT, AND, and OR. *See also* condition.

bound database

A named database in which all files (both dictionary and data files) are stamped with the database name. Binding ensures that the MicroKernel enforces the database's defined security, triggers, and referential integrity, regardless of the method (such as Btrieve or Scalable SQL) you use to access the data.

bound file

A data file associated with a single table in a specific named database. Scalable SQL automatically binds a file to a named database if the file meets any of the following criteria:

- Is part of a bound database
- Has a trigger
- Has a foreign key
- Has a primary key that is referenced by a foreign key

Scalable SQL requires a bound file to be bound to only one table; that is, no other tables in the database and no other databases can bind to the bound file.

browse

A mode of operation that allows you to query part of a database without making additions or changes. A browsing application lets you scroll forward and backward, examining the data before deciding what operation to perform.

b-tree

A multi-level or tree-structured index that provides a quick search path for data. Each branch and non-terminal node of the b-tree contains a range of possibilities within the index. During a search, the MicroKernel makes an evaluation at each node and chooses the path in the b-tree that falls within the range of its search. It follows this path through the b-tree until the desired data is located. This method is fast and efficient because the MicroKernel does not have to scan the entire index to find the requested information.

Btrieve

A complete navigational database management system, based on the architecture of the MicroKernel, designed for high-performance data handling and improved programming productivity.

Btrieve extended operation

An operation that returns or inserts multiple records on one Btrieve call (for example, Get Next Extended or Insert Extended).

Btrieve file

See data file.

Btrieve operation

A specific action (such as Delete, Create, or Get Equal) that manipulates a file.

Btrieve Requester

A program for the applicable DOS, OS/2, Win16, or Win32 environment that resides at a client machine and provides communication between the Btrieve server engine and a client application making Btrieve calls.

buffer

A storage area in memory that holds data temporarily.

cache

The area of memory that stores images of physical disk pages (or blocks of data). Using cache reduces the number of physical disk I/O requests and improves the MicroKernel's performance.

callback function

A function that is called from outside the code segment of a program. To yield time so that other tasks can run in the Windows 3.x environment, a Scalable SQL or Btrieve task must define and register a callback function with the respective engine. *See also* chaining callbacks.

candidate key

A column or group of columns whose column value (or collective column value) uniquely identifies each row in a table. Tables that have RI defined can have one or more candidate keys. For Scalable SQL, all candidate keys must be unique, non-null indexes. *See also* primary key, alternate key, referential integrity (RI).

Cartesian product join

See join.

cascade

A form of the delete rule. The cascade rule causes dependent rows to be deleted upon deletion of their parent rows. *See also* delete rule.

case sensitivity

An index attribute that determines how Scalable SQL evaluates uppercase and lowercase letters during sorting.

A key attribute that determines how Btrieve evaluates uppercase and lowercase letters during sorting.

See also index attribute and key attribute.

chaining callbacks

Linking more than one callback function together so that each callback is executed. A task can register as many callbacks as needed, but Scalable SQL and Btrieve keep track of only the last callback registered for that task. If the task needs to chain callbacks, the task is responsible for chaining each call to the previous callback from the current callback. *See also* callback function.

character set

A collection of characters and symbols, along with their computer representations, that are necessary to support a specific language (such as French or German).

character validation

See validation.

chunk

Any arbitrary portion of a record, specified by its offset and length. Btrieve enables applications to update and retrieve portions of very large records, called chunks, rather than the entire record.

clause

In a SQL statement, a substructure that contains a group of related items, such as keywords, values, subclauses, and conditions. For example, in the following SQL statement, the FROM clause contains the keyword FROM and a value (Person) for its associated table or view name:

```
SELECT * FROM Person
```

A clause may or may not be required in a SQL statement. For example, a SELECT statement requires a FROM clause.

Usually, a clause begins with keywords that are not the SQL command name. However, an entire SELECT statement can be a clause within another SQL statement, such as INSERT. Also, a restriction clause does not begin with a keyword, but contains conditions that define the search criteria for the data that the SQL statement affects.

See also keyword, restriction clause, SQL command, and SQL statement.

client

A task; also, a computer process that accesses the services and resources of other computer processes. In NetWare and Windows NT environments, a client typically is an application that accesses a server-based application. A client may also be a server-based application that accesses another server-based application. *See also* requester.

client/server configuration

A widely used computer system architecture in which machines designated as servers handle requests from numerous machines designated as clients. This configuration requires a requester module on the client, which communicates with a similar module on the server. Client/server configurations are necessary when controlled sharing of centralized data is required.

collation table

A table that maps a character set to an alternate collating sequence and changes the sort order of characters. *See also* alternate collating sequence and character set.

column

In Scalable SQL, a subdivision of the rows of a table. Columns define a vertical collection of values in a table. All the values in a particular column represent the same type of information.

column attribute

A characteristic assigned to a column and stored in the data dictionary. These characteristics include default value, edit mask, heading, null, and validation. *See also* default, edit mask, heading, null, and validation.

column list

A list of column names.

column name

The name you assign to a column in a table or a view when you create the table or view. *See also* qualified column name, table, and view.

column qualifier

A base table name, view name, or alias that unambiguously associates a column name with a table. A column qualifier is required when the column name is not unique in a SQL statement.

column value

The actual data stored in a column’s portion of a row. *See also* column, row, and table.

command

See SQL command.

command file

A user-defined file containing a sequence of commands that perform common operations.

commit

To save all the changes you have made to the database during a transaction. *See also* roll back and transaction.

computed column

A column that does not exist in the base table definition but is created in a view. A computed column can be a constant or it can be calculated from other columns in the view or from scalar functions. You can use a computed column in select lists. When you use a computed column in a restriction clause, it is referred to as an *expression* . *See also* expression.

conceptual design

The first phase of database design that involves determining and modeling database requirements.

concurrency

The ability of multiple tasks to access the same data simultaneously while preserving data integrity. *See also* concurrency controls.

concurrency controls

The methods the MicroKernel uses to resolve possible conflicts when two tasks attempt to access the same data at the same time. Concurrency controls in Scalable SQL include isolation levels for transactions, explicit record locks with XQLFetch, and passive control. Concurrency controls in Btrieve include passive control, record locking, and transaction control. *See also* explicit record locks, isolation level, and passive control.

concurrent transaction

A type of transaction that allows other transactions to take place simultaneously in different parts of the designated file. A concurrent transaction locks only the portions of the file (record or pages) that are accessed during that transaction, allowing other transactions to do the same in other portions of the file. *See also* exclusive transaction and implicit locks.

condition

An element of a restriction clause consisting of a condition operator and two expressions. The condition operator defines the criterion for comparing the expressions. You can combine multiple conditions using Boolean operators. *See also* Boolean operator, condition operator, expression, and restriction clause.

condition operator

An element of a restriction clause that defines the criterion for comparing two expressions. A condition operator can be either a range operator or a relational operator. *See also* range operator and relational operator.

configuration

The customization of various parts of a computer system for specific use. For example, you can use the Setup utility to configure MicroKernel options such as the number of open files and transaction durability.

constant

A nonchanging value that you specify in a SQL statement. For example, you can specify a numeric constant such as 3.14, a MONEY constant such as $40.00, or a string constant such as *George* .

You can compare a constant to another value (as in a condition), or you can apply an expression operator to a constant (as in a computed column definition that adds a numeric constant to another value). *See also* literal.

continuous operation

A MicroKernel feature that allows you to back up data files while they are open and in use. The MicroKernel opens the files in Read-Only mode to allow back up utilities to access the files' static images. The MicroKernel stores changes to the original files in temporary files called delta files. When the backup is complete, the MicroKernel automatically updates the original files with the changes stored in the delta files and deletes the temporary files as soon as all applications close the data file corresponding to that delta file.

control column

A column you specify in a GROUP BY clause to indicate how to group rows. Rows that have the same column value in the control column are grouped.

correlated subquery

A subquery that contains a WHERE or HAVING clause that references a table from the outer query's FROM clause. *See also* outer query and subquery.

Create Table right

A Scalable SQL security right that allows users or user groups to create new tables.

current row

The last row Scalable SQL returned on the previous XQLFetch operation. *See also* first row, last row, next row, and previous row.

cursor

An identifier obtained through a Scalable SQL function call that identifies a particular database operation to perform.

A named virtual table defined through the DECLARE CURSOR statement that allows controlled reading and writing of data through SQL.

See also cursor ID *and* view.

cursor ID

An integer value that references a cursor. You can use the SQL-level function XQLCursor to allocate a cursor ID. *See also* cursor *and* view.

cursor stability

An isolation level in which Scalable SQL locks portions (rows or pages) of a file during a transaction instead of locking the entire data file, thus allowing concurrent updates. Cursor stability is implemented using the concurrent transaction feature of the MicroKernel Database Engine. *See also* concurrency controls, concurrent transaction, Exclusive, implicit locks, and transaction.

cursor transaction

A type of nested transaction that enables you to selectively include files in transactions without subjecting other files to transaction control. With cursor transactions, you can diverge your views to see different stages of the same file simultaneously. You can use the cursor transaction in either Exclusive or Concurrent mode.

cycle

A reference path in which a parent table is its own descendant. *See also* descendant, parent table, reference path, and referential integrity (RI).

data abstraction

A DBMS can present data to an application in a form that is very different from the physical data structures. The form in which data is presented is called the data abstraction and determines the data model of the DBMS.

data administration statement

One of a group of SQL statements you can use to specify Scalable SQL session variables. These variables define blank replacement characters, isolation levels, file open modes, and file owner names. *See also* data control statement, data definition statement, data manipulation statement, and session variable.

data buffer

A Btrieve function parameter that you use to transfer various information depending on the operation being performed. A data buffer can contain all or part of a record, a file specification, and so forth.

data buffer length

A Scalable SQL or Btrieve function parameter that you use to specify how much data is in the data buffer parameter or how much data you expect to be in the data buffer when the function completes.

data compression

A method for reducing the space that a set of data occupies by encoding repeated information in a smaller form. The MicroKernel uses data compression to reduce the disk space of files that are appropriately configured.

data control statement

One of a group of SQL statements you can use to enable and disable security for a database, create users and user groups, and grant and revoke security rights. *See also* data administration statement, data definition statement, data manipulation statement, security, user, and user group.

data definition statement

One of a group of SQL statements you can use to create and delete dictionaries and indexes; create, modify, and delete tables; and define column attributes. *See also* data administration statement, data control statement, and data manipulation statement.

data dictionary

A set of tables, called *system tables* , that contain the complete description of a database. This description includes table and column names, data types, column attributes, index attributes, referential constraints, and security rights. The data dictionary is also called the *system catalog* . *See also* system tables.

data dictionary file

The physical file associated with a system table. In Scalable SQL, data dictionary files have a .DDF extension and are stored in a format compatible with the MicroKernel. *See also* system tables.

data file

A collection of related records stored on a disk. A data file is also referred to as a physical file, a MicroKernel data file, or simply a file. In Scalable SQL, a data file is presented as a table.

The MicroKernel creates and uses data files in which data is stored in different types of pages. These types include header, data, key, page allocation table (PAT), and variable-allocation table (VAT).

data file location

A directory path that Scalable SQL and Btrieve use to locate data files. Each maintains a list of directory paths in a data file path. When you specify a file name or a relative path, Scalable SQL and Btrieve locate the data file by appending the file name or relative path to the end of the appropriate directory path in the data file path. *See also* data file path.

data file path

A path that contains a list of data file locations (directory paths). When you search for a data file, Scalable SQL and Btrieve combine the data file name or relative path along with one of the directory paths in the data file path. When you create a data file, Scalable SQL and Btrieve combine the data file name or relative path along with the first directory path in the data file path. You specify the data file path when naming a database in the Setup utility or when logging in using paths in the XQLLogin function. *See also* data file location.

data manipulation statement

One of a group of SQL statements you can use to retrieve, insert, update, and delete data in tables; define transactions; create and delete views; and create, delete, and execute stored SQL statements. *See also* data administration statement, data control statement, and data definition statement.

data modification statement

One of a group of SQL statements you can use to add, change, or remove data.

data-only file

A data file in which no keys (and therefore no index pages) exist.

data page

See page.

data type

In Scalable SQL, an internal format for a column, such as INTEGER, DATE, or DECIMAL. The data type specifies what kind of data a column contains. For example, a column of data type DECIMAL stores numeric data with a fixed number of decimal places.

database

A set of one or more records or files that contain information on a related subject.

A collection of related information, such as employee data. *See also* relational database.

database administrator (DBA)

The individual or department responsible for managing databases and for granting user access rights.

database management system (DBMS)

A program or set of programs that create and manipulate database tables and associated files.

database name

A logical name for a database that allows you to refer to it without knowing its actual physical location. The Pervasive.SQL Setup utility allows you to manage database names.

database server

A database management system that runs on a server. A database server is a type of *back end* . *See also* back end and database management system (DBMS)**.**

DBA

See database administrator (DBA).

DBMS

See database management system (DBMS).

deadlock

A condition that occurs when each of two or more tasks is retrying operations on files, pages, or records that the other task has already locked. *See also* concurrency controls.

default

A preset value or option that a program chooses automatically when no other value is specified. For example, the default directory is the one in which you are currently working.

Default is also a column attribute specifying a value that is consistent with the data type of the column. Scalable SQL inserts this value in the column when no other value is specified. *See also* column attribute.

default mask

An edit mask that Scalable SQL defines for the data type of a column. *See also* edit mask, permanent mask, and temporary mask.

delete-connected

A table is delete-connected to another table if the deletion of rows in the first table causes the deletion of rows in the second table. Referential constraints determine whether a table is delete-connected.

Delete right

A Scalable SQL security right that allows users or user groups to delete rows from a table.

delete rule

A referential constraint that specifies how to handle dependent rows when someone attempts to delete their parent rows. The two forms that the delete rule can take are cascade and restrict. *See also* cascade and restrict.

delta file

When continuous operation begins, the MicroKernel creates a temporary data file, called a delta file, for each formatted data file. The MicroKernel records in the delta file any changes made to the data file while the backup is taking place. *See also* continuous operation.

dependent row

In a dependent table, a row whose foreign key value depends on a matching primary key value in the associated parent row.

dependent table

In a database with referential integrity (RI), a table that contains at least one foreign key. *See also* foreign key.

descendant

A dependent table on a reference path. A descendant can be one or more references removed from the path's original parent table. *See also* dependent table, parent table, and reference path.

descending

An index attribute that instructs SQL to collate an index in descending order. It is used in CREATE TABLE and CREATE INDEX statements, and in the ORDER BY clause in SELECT statements. *See also* index attribute.

A key attribute that instructs the MicroKernel to order key values in descending order. *See also* key attribute.

description file

A sequential file containing information necessary for certain Btrieve Maintenance utility operations.

development level

See SQL-level functions.

dictionary

See data dictionary.

dictionary file

See data dictionary file.

dictionary location

A directory containing a data dictionary.

dictionary name

A name stored in a data dictionary to identify a database element such as a table, view, index, or column.

directory

A disk structure that contains files. A directory may also contain subdirectories.

directory specification

A path that tells an application where to find or store information. The specification may include a server name, volume, drive letter, and directory path.

display mask

See edit mask.

DLL

See dynamic link library (DLL).

DOS

The DOS, PCDOS, or MSDOS operating system.

drop

To remove an item. For example, you can drop a table, view, index, or user group from a data dictionary.

duplicatable

An index attribute that instructs Scalable SQL to allow multiple rows to have the same column value in the indexed column. *See also* index attribute.

A key attribute specifying that multiple records in a file can have the same key value. *See also* key attribute.

duplicate key

A single key that identifies more than one record within the data file containing that key. The key’s definition specifies that more than one record in the file can contain the same key value.

dynamic link library (DLL)

A program library that contains related modules of compiled code. At runtime, the application reads the functions in the DLL. This process is called *dynamic linking* .

dynamic link routine

In OS/2 and Windows, a program that the operating system loads on demand (*dynamically*) and terminates automatically.

edit mask

A column attribute that specifies a display format for a column. For example, you can include commas in a numeric column to display the data in a more readable format. *See also* default mask, column attribute, permanent mask, and temporary mask.

edit mask literal

A character you can embed in an edit mask so that the character also appears in the data. For example, you can place dashes as literal characters in a social security column, or commas between groups of three numbers in a numeric column.

equal join

See join.

exception table

A table that contains all the orphan rows for a dependent table.

Exclusive

A file open mode. The user that opens a data file in Exclusive mode is the only user that can access that file. *See also* Accelerated, file open mode, Normal, Read-Only, and Verify.

exclusive isolation level

In Scalable SQL, an isolation level in which Scalable SQL causes the MicroKernel Database Engine to use the entire data file as the locking unit for transactions. The exclusive isolation level is comparable to using exclusive transactions in Btrieve. *See also* concurrency controls, cursor stability, Exclusive, exclusive transaction, isolation level, and transaction.

exclusive transaction

A type of transaction that causes the MicroKernel to lock the entire file when you insert, update, or delete a record. The file remains locked until the task ends or aborts the transaction. *See also* concurrent transaction and implicit locks.

explicit record locks

A type of concurrency control in Scalable SQL and Btrieve in which you explicitly lock records by specifying a lock bias value on an XQLFetch call or by including the lock request with a Btrieve operation code. Explicit record locks lock the records for update and delete operations but do not allow you to roll back the operations. These locks have no effect within a transaction. *See also* concurrency controls and implicit locks.

explicit transaction processing

A type of transaction processing in which you must indicate the beginning of a transaction by issuing a START TRANSACTION statement. You end the transaction by issuing a COMMIT WORK or ROLLBACK WORK statement.

expression

An element of a condition in a restriction clause, or an element in a select list. An expression can be one of the following items:

- A constant value
- A column name (qualified or not)
- A scalar function

- A computed column that uses one or more of the above

See also computed column, condition, constant, column name, restriction clause, scalar function, and select list.

expression operator

An element of a condition that connects two expressions to form another expression. An expression operator can be an arithmetic operator or a string operator. *See also* arithmetic operator, Boolean operator, condition, expression, and string operator.

extended file

The MicroKernel creates an extended file when it processes an operation, such as inserting a record into a file, that causes a file to add a page that extends the file size beyond the operating system limit.

extended operation

See Btrieve extended operation.

extension file

A physical file logically concatenated with a base file when the MicroKernel creates an extended file. The MicroKernel treats the base file and extension files as one large file. *See also* extended file.

field

In Btrieve, the term *field* has been used historically to refer to portions of a record that have been designated as segments of a key. Technically speaking however, Btrieve records contain no fields.

file

A collection of records stored on a disk. A file is sometimes also referred to as a *physical file* .

file control record (FCR)

A page in a data file that contains information1 about the file, such as the file size and page size. The first two pages in any data file are FCR pages.

file definition

A data file description stored in a data dictionary.

file-level locking

See Exclusive.

file open mode

A method of opening a file that places restrictions on how that file can be accessed. The file open modes are Accelerated, Exclusive, Normal, Read-Only, and Verify. *See also* Accelerated, Exclusive, Normal, Read-Only, and Verify.

file owner name

A password that protects data files from unauthorized access by MicroKernel applications. In Scalable SQL, you can assign a file owner name to a file when you use a CREATE TABLE statement. In Btrieve, you can assign an owner name using the Maintenance utility or by implementing a Set Owner operation.

file preallocation

See preallocation.

file sharing

File sharing is the ability of multiple users or tasks to operate on the same file simultaneously. With the workstation engine, the MicroKernel provides two file sharing modes:

- Single Engine File Sharing (SEFS) mode—Only users or tasks running under the same MicroKernel can access or maintain information stored in a shared file. Users or tasks running under different MicroKernels have no access to the shared file.

  By default, the workstation MicroKernel uses SEFS mode if the shared file resides on a local drive. (A local drive is physically located on the same machine running the MicroKernel.)

- Multi-Engine File Sharing (MEFS) mode—Users or tasks running under different MicroKernels can access or maintain the shared file.

  The workstation MicroKernel uses MEFS mode by default if the shared file resides on a remote drive. (A remote drive can be a drive on a file server or a drive on a peer machine.)

File sharing is not restricted to these default values.

filter

A restriction you can use when retrieving records with a Btrieve extended operation.

first row

An absolute row position based on the highest index position of the rows in the view. The first and last rows in the view are absolute positions that remain in effect until you compile a new SQL statement for the cursor ID or release the cursor ID. Your first XQLFetch operation must position the cursor to either the first or last row in the view, based on the current sort path. *See also* current row, last row, next row, previous row, and sort path.

fixed-length data type

A data type with a designated stored length that does not vary from row to row within a table. *See also* variable-length data type.

fixed-length record

A record that contains no portions of variable length. *See also* variablelength record.

foreign key

A column or group of columns that references a primary key. The foreign key can be in the same table as the primary key or in a different table. A foreign key cannot contain null column values, and the columns it includes must match the primary key columns in type, length, and order. *See also* primary key and referential integrity (RI).

free space threshold

A mechanism the MicroKernel uses to determine whether to add data to an existing variable page or to create a new one. A higher free space threshold reduces fragmentation of variable-length records across several pages but uses more disk space. The MicroKernel also stores compressed records (even when fixed-length) on variable pages.

front end

An application that provides an interface between a user and a back end (or engine). For example, SQLScope is a front end to the Scalable SQL back end. *See also* back end.

front-end result set

A cache on the client computer that holds the results of a query.

group

See user group.

group aggregate function

One of a group of SQL functions you use to return a single result for a given set of column values. The group aggregate functions are AVG, COUNT, MIN, MAX, and SUM. You can use group aggregate functions in a select list or a HAVING clause.

heading

A column attribute that specifies a column name different from the name you defined for the column in the dictionary. You specify column headings in a CREATE VIEW statement. *See also* column attribute.

hierarchical data model

A database access method in which the data is represented in the form of a set of tree structures (hierarchies), and the operators provided for manipulating such structures for traversing hierarchic paths up and down the trees. *See also* relational data model and navigational data model.

implicit locks

The type of locking in which Scalable SQL automatically locks rows, pages, or files according to the isolation level you specify for your transactions. *See also* concurrency controls, explicit record locks, isolation level, and transaction.

The type of locking in which the MicroKernel automatically locks a record, page, or file according to the type of operation and transaction you are using. In general, the MicroKernel locks a file during an exclusive transaction and a page or record during a concurrent transaction. *See also* concurrent transaction and exclusive transaction.

index

A column or combination of columns used to sort a table in a particular order and to optimize searches on particular values. *See also* segmented index.

A key or group of keys that the MicroKernel uses to sort a file. *See also* key.

index attribute

A characteristic assigned to an index and stored in the data dictionary. The index attributes are: case-sensitivity, modifiability, null value indexing, segmentation, sort order, and uniqueness.

See also case sensitivity, modifiable, null value indexing, segmented, sort order, and uniqueness.

index balancing

The process of searching for available space in sibling index pages when a given index page becomes full, and then rotating values from the full page into the pages that have space available.

index page

See page.

index path

A logical ordering of rows in a table based on indexed column values.

index segment

One column of an index that includes multiple columns. *See also* index and segmented index.

inner query

See subquery.

Inscribe

A development technology that helps you create and run scripts for use with Scalable SQL's stored procedure stored procedure feature. Inscribe uses Softbridge Basic Language (SBL), a programming language compatible with Visual Basic, to create scripts.

Insert right

A Scalable SQL security right that allows a user or user group to add new rows to a table.

insert rule

A referential constraint that specifies how to apply inserts to dependent tables. The insert rule is always *restrict* , which prevents adding a row to a dependent table if the row contains a foreign key value not contained in the parent table.

integrity control

A method of ensuring that the data in a file is complete and accurate. Scalable SQL uses referential integrity, security, and triggers to guarantee data file integrity. The MicroKernel uses shadow paging and concurrency controls to guarantee data file integrity. *See also* concurrency controls, referential integrity (RI), security, and shadow paging, and triggers.

International Sorting Rules (ISR)

A syntax for specifying an external collation sequence for data files, used in translation.

internetwork

Two or more separate, independent networks joined by a communications bridge. The bridge enables the file servers in separate networks to access each other’s data.

isolation level

The level of data locking that Scalable SQL employs to provide isolation from other changes to the data during a transaction. The locking level determines the degree to which a task’s transaction affects the ability of other tasks to read and update data in the same data file. Scalable SQL supports two levels of isolation: cursor stability (in which the locking unit is a row or page) and exclusive (in which the locking unit is a file). *See also* concurrency controls, cursor stability, Exclusive, implicit locks, and transaction.

join

A basic relational operation between two or more tables that combines data in the tables based on a comparison of values in specified columns of those tables. Following are the types of joins:

- Equal join—a join that occurs when you specify the two join columns as being equal.
- Nonequal join—a join that is based on a nonequal comparison of two join columns. The types of nonequal comparisons are less than, greater than, less than or equal, and greater than or equal.
- Null join—a join that allows you to retrieve each row from one table (on the left), regardless of whether there is a corresponding row in the table to which you are joining (on the right).
- Cartesian product join—a join that associates each row in one table with each row in another table.
- Self join—a join in which you specify a table name in the FROM clause more than once.

See also table and view.

join condition

An expression containing a comparison between columns from two different tables. *See also* expression, restriction clause, and restriction condition.

key

A column or group of columns on which a table’s referential integrity (RI) constraints are defined. The types of keys are primary and foreign. A *primary key* is a column or group of columns that uniquely identifies each row in a table. A *foreign key* is a column or group of columns that references a primary key. *See also* foreign key, primary key, and referential integrity (RI).

A group of bytes (or multiple groups) that is characterized by offset and length (that is, by physical location in a record) and provides a means of direct access to a data value. In addition, a key provides a means of dynamic sorting of the records within a data file. *See also* index and segmented key.

key attribute

A characteristic assigned to a key. The key attributes are: alternate collating sequence, case-insensitive, descending, duplicatable, manual, modifiable, null, repeating-duplicatable, segmented, and supplemental. *See also* case sensitivity.

key buffer

A Btrieve function parameter that usually contains the value of a key identified by the key number parameter.

key number

An identifier associated with a specific key in a data file.

key-only file

A data file in which no data pages exist; that is, all records are stored on index pages.

key segment

One of the groups of bytes that a segmented key includes. *See also* key and segmented key.

key type

See data type.

keyword

A word that is part of the SQL language. Although keywords appear in uppercase letters in the documentation, they are not case-sensitive. *See also* clause, SQL command, and SQL statement.

last row

An absolute row position based on the lowest index position of the rows in the view. The first and last rows in the view are absolute positions that remain in effect until you compile a new SQL statement for the cursor ID or release the cursor ID. Your first XQLFetch operation must position the cursor to either the first or last row in the view, based on the current sort path. *See also* current row, first row, next row, previous row, and sort path.

linked-duplicatable

A key attribute that instructs the MicroKernel to store on an index page a key value extracted from the first record of a duplicatable key. Then, rather than storing the second duplicate for the key on the index page, the MicroKernel places a pointer to the second duplicate key at the end of the first duplicate key's record. The MicroKernel places a pointer to the third duplicate key at the end of the second duplicate key's record, and so on. *See also* key attribute and repeating duplicatable.

literal

A literal is an exact representation of a data value, as opposed to a reference to a column or expression containing that value. A literal can be any valid data type, such as a character literal ('Jones'), a numeric literal (123.4 or 6.02E+23), or a date literal (12/06/95).

See also edit mask literal**.**

loader

A program that loads another program into memory.

local client

Two clients are local clients relative to each other if they run under the same MicroKernel. A given client can be both local relative to some clients (those running under the same MicroKernel) and remote relative to other clients (those running under different MicroKernels) at the same time.

location transparency

A database feature that enables a user to access a database without specifying its physical location.

lock

A mechanism that prevents other tasks from changing the data you currently have locked. *See also* concurrency controls, explicit record locks, implicit locks, isolation level, no-wait lock, and wait lock.

locking unit

The amount of data Scalable SQL blocks from other tasks until your transaction is complete. Scalable SQL uses locking to prevent other tasks from changing the data you are currently changing. *See also* cursor stability, Exclusive, isolation level, and transaction.

logger

A task that calls the xShareSessionID function for its login session. xShareSessionID allows a task to flag its login session as shareable so that other tasks can share the resources associated with this login. *See also* session and sharer.

logging in

The act of connecting to a data dictionary. If security is enabled, logging in identifies the user so that Scalable SQL can enforce the appropriate security rights. Users log in using a task (an instance of an application).

Scalable SQL allows a task to have multiple logins or share the login session of another task. Only one login session can be active within a particular task. Thus, only the cursor(s) defined for the current login session is active. A Scalable SQL security right required for logging in to a data dictionary. All users must log in to a dictionary before accessing data. *See also* logger, session, and sharer.

logical design

The second phase of database design that involves creating tables, columns, and keys based on the relational model that meet the requirements identified in the conceptual design.

logical operator

An operator such as AND, OR, or NOT used in an expression that yields a true or false value.

logical unit of work

See transaction.

login identifier

A unique number that identifies a particular application operating at a workstation. Scalable SQL assigns a login identifier to each application that logs in.

Login right

A Scalable SQL security right required for logging in to a data dictionary. All users must log in to a dictionary before accessing data.

login session

See session.

log key

The key that the MicroKernel uses to uniquely identify each record in a file for transaction logging purposes. If a file does not contain a unique (non-duplicatable key), the MicroKernel cannot track the file in the log. Using the Setup utility, you can configure the MicroKernel to add a unique *system-defined log key* to files during creation.

log segment

A file that comprises part of the MicroKernel transaction log. All log segments have a.LOG extension. The MicroKernel names log segments with consecutive, 8-character hexadecimal names, such as 00000001.LOG, 00000002.LOG, and so on.

manual key

See null key.

many-to-many relationship

Occurs when many rows in one table can relate to many rows in another table.

mask

See edit mask.

mask literal

See edit mask literal.

Master user

The user who has all rights to the dictionary after security is enabled. The user name *Master* is case-sensitive.

MicroKernel Database Architecture

The unique modular framework for Pervasive Software’s database engine products. The architecture consists of access modules such as Btrieve and Scalable SQL, both of which plug into the MicroKernel. This architecture allows applications that use different data access methods to share common data.

MicroKernel Database Engine

The foundation of the MicroKernel Database Architecture. Provides low level database functions that include physical data management, data caching, transaction processing, and data integrity enforcement.

modifiable

An index attribute that determines whether you can modify an index column value after Scalable SQL stores the associated row. *See also* index attribute.

A key attribute that determines whether you can modify a key value after Btrieve stores the associated record. *See* *also* key attribute.

multi-byte character set

A character set in which characters may be represented in an arbitrary number of bytes.

Multi-Engine File Sharing (MEFS)

See file sharing.

multithreading

Concurrent, or interleaved, execution of multiple threads of execution.

named database

A database with a logical name that allows users to identify it without knowing its actual location. When you name a database, you associate that name with a particular dictionary directory path and one or more data file paths. When you log in to Scalable SQL using a database name, Scalable SQL uses the paths to find the database's dictionary and data files. *See also* database name.

named index

In Scalable SQL, an index you name using the CREATE INDEX command. You can delete a named index, but you cannot delete an unnamed index (an index you create using a WITH INDEX clause in a CREATE TABLE statement).

native character set

The character set in which text submitted to and generated by the Scalable SQL engine is assumed to be represented. This set is defined by the host environment of the system on which it is running.

navigational data model

The data access method used to navigate up, down, and sideways through data records. This model provides direct control and allows a developer to optimize data access based on knowledge of the underlying structure of the data. Btrieve is a navigational database engine. *See also* hierarchical data model and relational data model.

NDS

NetWare Directory Services.

nested query

See subquery.

nested transaction

A type of transaction that enables you to divide complex transactions into smaller subtransactions. With nested transactions, you can partially roll back changes in a transaction by committing or undoing subgroups of transactions while the main transaction remains unaffected. You can use the nested transaction in either Exclusive or Concurrent mode.

NetWare Loadable Module (NLM)

A program that runs on a NetWare 3.*x* or 4.*x* server. You can load or unload NLMs while the server is running. NLMs become part of the operating system and access NetWare services directly.

network administrator

The individual or department responsible for the communications link between a user's computer and network servers.

next row

Based on the current sort path, the row that follows the set of rows returned on the previous XQLFetch operation for all XQLFetch options except Option 0 (Fetch Current). If you use Option 0 to fetch multiple rows, the next row for the subsequent fetch operation is immediately adjacent to the current row. *See also* current row, first row, last row, previous row, and sort path.

NLM

See NetWare Loadable Module (NLM).

nonequal join

See join.

non-null index

An index that does not include null column values. *See also* null.

Normal

The default file open mode. Using a server MicroKernel, Normal mode allows shared read/write access to data files. In Normal mode, the MicroKernel performs its standard integrity processing when it updates the data files.

In the Windows environment using the 6.15 workstation MicroKernel, Normal mode is the equivalent of Single Engine File Sharing (SEFS) mode if the shared file resides on a local drive. A local drive is one that is physically located on the same machine that is running the local MicroKernel. If the shared file resides on a remote drive, Normal mode is the equivalent of Multi-Engine File Sharing (MEFS) mode with the 6.15 local MicroKernel. A remote drive can be a drive on a NetWare server or a drive on a peer machine (if in a peer-to-peer networking environment).

See also Accelerated, Exclusive, file open mode, file sharing, Read-Only, and Verify.

normal form

A set of criteria to measure the level at which a database conforms to certain basic rules of data organization.

normalization

The process of transforming your database into a particular normal form by applying a set of rules.

no-wait lock

A type of explicit record lock in which the MicroKernel returns control to your task if it cannot lock the record for any reason (for example, because another task has already locked that record).

In Scalable SQL, you request a no-wait lock by specifying a lock bias value on an XQLFetch call.

In Btrieve, you request a no-wait lock by specifying a lock bias value on a Btrieve operation code.

See also explicit record locks and wait lock.

null

Specifies an unknown column value, or specifies that the column value is not applicable.

null join

See join.

null key

A key column that can be a user-defined null character. For null keys, the MicroKernel does not include a record in the index if each byte of the record's key matches the null column value.

A key field that can be a user-defined character. The MicroKernel allows two types of null keys: any-segment (called a manual key in earlier Btrieve versions) and all-segment (simply called a null key in earlier Btrieve versions).

An any-segment null key does not include a particular record in the index if the value of any key segment of that record matches the null value.

An all-segment null key only excludes a particular record from the index if the value of all key segment of that record matches the null value.

null value indexing

An index attribute that determines whether Scalable SQL includes those columns that contain the predefined null value in the index. *See also* index attribute.

A key attribute that determines whether Btrieve includes those fields that contain the predefined null value in the index. *See also* key attribute.

object-oriented data model

The data access method in which data is represented as objects that represent real world entities such as students, courses, and grades. These objects consist of both data and data functions, such as creating or deleting objects.

one-to-one relationship

Occurs when one row can relate to only one instance of a row in another table.

one-to-many relationship

Occurs when one row can relate to many rows in another table.

Open mode

See file-level locking.

operand

An input value for a restriction clause, condition, expression, or scalar function. Depending on the context, an operand may be the result of a previous operation.

An operand is also a value to which an operation is applied. *See also* condition, expression, restriction clause, and scalar function.

operation

A specific action that manipulates a data file (such as Delete, Create, or Get Equal). An operation is performed when an application calls one of the Btrieve functions.

operator

Part of a restriction clause, condition, or expression. The types of operators are Boolean operators, condition operators (which include relational operators and range operators), and expression operators (which include arithmetic and string operators). *See also* arithmetic operator, Boolean operator, condition operator, expression operator, range operator, relational operator, and string operator.

orphan row

In a dependent table, a row with a foreign key value that does not have a corresponding parent key value.

outer join

See join.

outer query

The main query of a SQL statement that contains a subquery. *See also* correlated subquery and subquery.

owner name

See file owner name.

page

A unit of a data file. A page is the smallest unit of storage that the MicroKernel moves between main memory and disk. A page contains a multiple of 512 bytes (up to 4,096 bytes).

The MicroKernel uses the following types of pages:

- Data page—contains fixed-length records (or the fixed-length portions of variable-length records)
- Index page—contains key values and pointers to the associated records for those values (which reside on a data page)
- Variable page—contains variable-length portions of records

See also data file, fixed-length record, key, and variablelength record.

Page Allocation Table (PAT)

Part of the MicroKernel’s internal implementation for tracking pages in a file. *See also* shadow paging.

page-level locking

A level of data locking that causes the MicroKernel to lock individual data pages during a transaction instead of locking the entire file. This permits multiple users to access different pages in the data file from within a transaction. In Scalable SQL, page-level locking is called *cursor stability* . *See also* Exclusive and transaction.

parameter

An item of information that a program, utility, or API may need in order to perform a particular operation. *See also* substitution variable.

parent row

In a parent table, a row whose primary key value currently matches a foreign key value. *See also* dependent row.

parent table

In a database with referential integrity (RI), a table containing a primary key referenced by a foreign key. *See also* dependent table.

parsing

The mechanism the SQL engine uses to analyze the SQL statements that are passed to it and determine what internal operations it must perform to execute the SQL statement.

passive control

A type of concurrency control in which your task does not perform any type of locking. If another task modifies a record since you originally fetched it, you must fetch the record again before you can perform an update or delete operation. *See also* concurrency controls.

path

The components that uniquely identify a file or directory. For local files, these components might include a drive letter, directory levels, and a file name. For network files accessed from a client, these components might include the server name, volume, directory path, and file name.

peer-to-peer networks

Enable machines to access files that reside on other machines. Multiple users running applications on one machine can access database files stored on another machine.

permanent mask

An edit mask that you store in the dictionary that becomes part of the column’s data definition. You can specify a permanent mask using a SET MASK statement. *See also* default mask, edit mask, and temporary mask.

physical design

The third phase of database design that involves specifying data types, indexes, CREATE statements, and sizes of the physical files representing the tables.

position block

A MicroKernel data file handle.

positioning

A record’s location relative to the locations of the records preceding and following it. The two types of positioning are physical and logical. Physical positioning refers to the three relevant record locations in the data portion of a data file; logical positioning refers to the three relevant record locations in the key index of a data file.

preallocation

A method for allocating a specific amount of disk space for a file when you create the file. This space is reserved for future expansion of the file.

pre-imaging

Storing the image of a data file page before updating a record on that page. Pre-6.0 engines use preimaging to provide recovery capabilities in case a file is damaged or a system failure occurs during an update to the file.

previous row

Based on the current sort path, the row that precedes the set of rows returned on the previous fetch operation for all XQLFetch options except Option 0 (Fetch Current). If you use Option 0 to fetch multiple rows, the previous row for the subsequent fetch operation is immediately adjacent to the current row. Normally, you use Option 0 to fetch only the current row. *See also* current row, first row, last row, next row, and sort path.

primary file

The original part of a partitioned file before a Btrieve Extend operation was performed.

primary key

A column or group of columns whose column value (or collective column value) uniquely identifies each row in a table. Primary keys must be unique, non-null indexes. *See also* foreign key and referential integrity (RI).

procedure

See stored procedure.

procedure owned cursor

A SQL cursor defined inside a stored procedure.

procedure owned variable

A SQL variable you define inside a stored procedure.

projection

See column list.

Public group

A special user group you can use to specify the minimum set of security rights every user has when accessing a dictionary on which security is enabled. You cannot delete the Public group. All users are automatically members of the Public group and cannot be deleted from it.

qualified column name

A column name preceded by a column qualifier. A qualified column name allows you to differentiate between column names that are repeated in different tables. *See also* column name and column qualifier.

query

A request for information from a database. Through SQL, you use a SELECT statement to specify the query.

range

A column attribute that defines a range of valid values for a column.

range checking

See validation.

range operator

A type of condition operator that compares the value of an expression to a specified range of values. Examples of range operators for SQL statements are CONTAINS and IS NULL. *See also* condition, condition operator, expression, and restriction clause.

Read-Only

A file open mode that does not allow you to insert, update, or delete records. *See also* Accelerated, Exclusive, file open mode, Normal, and Verify.

read-only view

A view in which you cannot insert, update, or delete rows.

record

A set of logically related data items in a MicroKernel data file. For example, a record might contain an employee's name, address, phone number, rate of pay, and so forth. Multiple records make up a data file.

record locking

A type of concurrency control that enables an application to lock the record it is accessing within a file. Other users can read the record, but no other user can lock, update, or delete the record until the application that holds the lock releases it.

reference

A foreign key referring to a primary key. *See also* foreign key, primary key, and referential integrity (RI).

reference path

A particular set of references between dependent and parent tables. *See also* referential integrity (RI).

References right

A Scalable SQL security right that allows users or user groups to create foreign keys that refer to the primary key of a table.

referential constraints

A set of rules that define the relationships between tables in a database.

referential integrity (RI)

The assurance that when a column (or group of columns) in one table refers to a column (or group of columns) in another table, changes to those columns are synchronized. If you define RI on a bound database, the MicroKernel enforces the database's defined security, triggers, and referential integrity, regardless of the method (such as Btrieve or Scalable SQL) you use to access the data. *See also* bound database.

relation

See table.

relational database

Related data that is systematically organized into a set of tables.

relational database management system (RDBMS)

Organizes and provides access to a relational database. A program for general-purpose data storage and retrieval.

relational data model

The data access method in which data is represented as collections of tables, rows, and columns. The relational model insulates the developer from the underlying data structure and presents the data in a simple table format. *See also* navigational data model and hierarchical data model.

relational operator

A type of condition operator that compares a column value with either another column value or a constant. The relational operators are less than (<), greater than (>), equal to (=), less than or equal to (<=), greater than or equal to (>=), and not equal to (!= or <>). *See also* condition, condition operator, and expression.

relational primitive

One of a group of lower-level Scalable SQL API functions that provide a relational link between a database application and data files. Relational primitives let you perform relational database operations using function calls from a standard programming language such as C, BASIC, COBOL, or Pascal.

relative path

A partial path that Scalable SQL and Btrieve append to a data file location in order to locate a data file. *See also* data file location.

remote client

Two clients are remote clients relative to each other if they run under different MicroKernels. A given client can be both local relative to some clients (those running under the same MicroKernel) and remote relative to other clients (those running under different MicroKernels) at the same time.

repeating duplicatable

A key attribute that instructs the MicroKernel to store key values with the data and in the index. *See also* key attribute and linked-duplicatable.

requester

A program that resides on a client machine and passes requests from a client application to a server-based engine. *See also* client.

requester interface

A program that resides on a client machine and passes requests from an application to a requester.

restrict

A form of the delete, insert, or update rule:

- The delete restrict rule prevents a row in a parent table from being deleted if that row is a parent row for a foreign key in another table.
- The insert restrict rule prevents a row from being added to a dependent table if the foreign key value the row contains does not have an equivalent primary key value in the applicable parent table.
- The update restrict rule prevents a row in a dependent table from being updated if the foreign key value that row contains does not have an equivalent primary key value in the applicable parent table.

See also dependent table and parent table.

restricting rows

Limiting the number of rows with which you will work by setting conditions for specific columns. For example, you can restrict rows in a mailing list to retrieve the rows for only one particular state.

restriction clause

Part of a SQL statement that defines one or more search criteria to qualify the data that the SQL statement affects. A restriction clause defines either a restriction condition or a join condition. You can include a restriction clause in the following places:

- The WHERE or HAVING clause of a SELECT statement or SELECT clause
- The WHERE clause of an UPDATE or DELETE statement

See also join condition and restriction condition.

restriction condition

Criteria specified in a restriction clause to compare an expression that references a column value to either a constant or another expression that references a column value in the same table. *See also* expression, join condition, and restriction clause.

result set

A set of rows retrieved from one or more tables or views during a query.

result table

See result set.

RI

See referential integrity (RI).

roll back

Aborting a transaction and undoing all the changes made to a file during the transaction, thus restoring the database to the state it was in before the transaction began. *See also* commit and transaction.

roll forward

Recovering changes made to a data file between the time logging is initiated and a system failure occurs. *See also* archival logging.

roll in

Writing to an original data file all the changes made to the corresponding temporary file during the continuous operation backup period. When the backup is complete, the MicroKernel automatically updates the original file with the changes and deletes the temporary file. *See also* continuous operation.

router

A program through which a client application communicates with a database or gateway. The router enables a logical connection between a client and a database or gateway. Once this connection is established, the client application uses the router to send SQL or Btrieve requests to the database or gateway and to receive results.

row

A set of logically associated columns in a view. For example, a row may contain some or all columns from a single table. Alternatively, it may contain columns from several tables that are joined to form a view.

savepoint

In a SQL transaction, you can define additional markers called savepoints. Using savepoints you can undo changes after a savepoint in a transaction and continue with additional changes before requesting the final commit or abort of the entire transaction.

scalability

The ability of a product to operate in computing environments of varying capabilities. No change to source code or relinking is necessary to scale from workstation to client/server operation.

Scalable SQL

Pervasive Software's relational database that allows users to run applications designed to manage shared data files.

scalar function

One of a group of SQL functions you can use to perform an operation on a particular data type, such as returning the length of a string, truncating a numeric value, or returning the day or hour value. You can specify scalar functions in statements that allow computed columns in expressions.

scripting

A programming language for developing routines that you can execute directly or within an application. For example, the scripting language in the Scalable SQL product is compatible with Visual Basic.

security

A means of protecting a database by limiting access to it. You can define data security using SQL statements or Btrieve owner names. For example, you can allow a user to see only certain columns in a table, or you can allow the user to see all the columns in a table but not update them.

security group

See user group.

Security right

A right that defines what operations a user can perform on a database. The Login and Create Table rights determine whether the user can log in to the dictionary and create tables. The *access rights* determine the user's ability to access tables in a dictionary. The access rights are Select, Update, Insert, Delete, Alter, References, and All. *See also* access right, All right, Alter right, Create Table right, Delete right, Insert right, Login right, References right, Select right, and Update right.

segmented

An index or key attribute that indicates whether the index or key is segmented (whether it consists of a group of columns combined into a single index or key). *See also* index, index attribute, key, key attribute, segmented index, and segmented key.

segmented index

An index consisting of multiple columns. The columns can be of different data types and need not be adjacent in the table. *See also* index.

segmented key

A key consisting of multiple groups of bytes characterized by offset and length. Key segments can be of different extended key types and do not have to be contiguous in the record. *See also* key.

select list

A list of select terms. *See also* select term.

Select right

A Scalable SQL security right that allows users or user groups to query tables for information.

select term

A column, computed column, or group aggregate function that a SELECT statement retrieves or computes.

self join

See join.

self-referencing table

A table that is its own parent table. In other words, the table contains a foreign key that references its primary key. *See also* parent table and referential integrity (RI).

Sequenced Packet Exchange (SPX)

A Novell communication protocol that monitors network transmission to ensure successful delivery. SPX runs on top of Novell's Internetwork Packet Exchange (IPX) protocol.

server

A computer that is on a network and provides services to client applications.

server-based MicroKernel

The server-based version of the MicroKernel Database Engine.

See also MicroKernel Database Engine**.**

session

The period during which a task is logged in to a database. A task can create multiple login sessions by logging in more than once, or the task can share the login session of another task. *See also* logger and sharer.

session cursor

A SQL cursor you define outside of a procedure.

session identifier

An integer value that identifies a Scalable SQL login session. The integer value is a value from 0 to the maximum number of concurrent login sessions that Scalable SQL allows. *See also* session.

session variable

A SQL variable you define outside of a stored procedure.

shadow paging

In a data file, the process of changing a copy of a page (called a shadow page) instead of changing the current page. When the changes are committed, the shadow page becomes the current page and the current page is ready for reuse. *See also* Page Allocation Table (PAT).

sharer

A task that shares a logger's login session. A sharer does not need to be logged in to share the logger's login session. However, the sharer can also have its own login session. *See also* logger and session.

sibling networks

Two or more equal networks branching off the same node in an internetwork.

significant blanks

Blanks that appear at the end of a string and that Scalable SQL treats as part of the data when displaying the string, determining its size, or comparing it to another string.

Single Engine File Sharing (SEFS)

See file sharing.

sort file

See temporary sort file.

sort order

An index attribute that determines how Scalable SQL sorts index column values. By default, Scalable SQL sorts index column values in ascending order (from smallest to largest).

A key attribute that determines how the MicroKernel sorts key values.

See also index attributekey attribute.

Sort order is also the order by which you sort a view when using an ORDER BY clause. You can sort the view by one or more columns, and the columns can be either index columns or nonindex columns. For example, if you sort a view in ascending order using a string column, the rows that contain a column value beginning with the letter *A* appear first in the view.

sort path

The ordering path of the view, which the ORDER BY clause or the optimizer determines if you did not specify an ORDER BY clause.

SPX

See Sequenced Packet Exchange (SPX).

SQL

See Structured Query Language (SQL).

SQL command

The initial keyword(s) in a SQL statement. For example, in the following SQL statement, SELECT is the SQL command:

```
SELECT * FROM Faculty
```

See also clause, keyword, and SQL statement.

SQL-level functions

A development level that serves as a high-level interface between your application and the MicroKernel Database Engine. The SQL-level functions allow your application to use SQL statements to create, access, and modify relational database systems. You can invoke the SQL-level functions from a standard programming language, such as C/C++, BASIC, COBOL, or Pascal. Examples of SQL-level functions are XQLLogin, XQLCompile, and XQLFetch.

SQL statement

A complete Scalable SQL request that contains a SQL command, any required values, and any required (or optional) clauses. *See also* clause, keyword, and SQL command.

statement

See SQL statement.

stored procedure

A group of logically associated programming steps you can invoke with a single SQL statement. Once invoked, SQL stored procedures are executed in their entirety without internal communication between a host language program and the SQL engine.

stored view

A view whose definition you have placed in the data dictionary for later retrieval. In most statements, you can use the name of a stored view in place of a table name.

string

A series of characters (as opposed to a number), or a category of data types used to store strings.

string operator

An expression operator you can apply to fixed-length string columns to form a computed string column. The string operators are append (*), concatenate without spaces (+), and concatenate with spaces (++). *See also* expression operator.

Structured Query Language (SQL)

A standard set of statements you can use to manage information stored in a database. For example, these statements allow you to retrieve, insert, update, or delete data. The types of SQL statements are as follows:

- Data administration
- Data control
- Data definition
- Data manipulation

See also data administration statement, data control statement, data definition statement, and data manipulation statement.

subquery

A SELECT statement that is contained within the WHERE clause of a SELECT, UPDATE, or DELETE statement, or in the HAVING clause of a SELECT statement. A subquery is also called an *inner query* or a *nested query* . *See also* correlated subquery and outer query.

substitution variable

In a SQL statement, an identifier that instructs Scalable SQL to substitute a value in place of the identifier when executing the statement. The symbol @ identifies a substitution variable in a SQL statement.

supervisor

The person responsible for the administration and maintenance of a network, a database, or both. A supervisor has access rights to all volumes, directories, and files.

supplemental index

In Btrieve 5.x, an index added to a file after the file was created. In 5.x, you can delete a supplemental index, but you cannot delete an index that was created when the file was created. In Btrieve 6.0 or later, supplemental indexes are referred to as *repeating duplicatable indexes* .

symmetric multiprocessing (SMP)

SMP systems consist of multiple CPUs within a single machine that share a common address space. The operating system can allocate processes and threads to the various processors. Typical SMP systems consist of two to six CPUs.

SMP support in an application enables the application to use multiple processors. This requires that the application be multi-threaded so that the different threads of the application can be executed on separate processors.

system catalog

See data dictionary.

system data

A set of bytes the MicroKernel can add to files for logging purposes. System data includes a non-duplicatable (unique) key that the MicroKernel uses to track individual records for logging.

system-defined log key

A key the MicroKernel can add to files for transaction logging purposes. System keys are 8 bytes long and non-duplicatable (unique). Using the Setup utility, you can configure the MicroKernel to add system data, including a system-defined log key, to files upon creation. System-defined log keys are necessary for the MicroKernel to log files that do not contain at least one unique (non-duplicatable) key.

system tables

In a data dictionary, tables that contain descriptions of the database. For example, these tables contain information about your data, tables, views, and system security, as well as referential integrity (RI). Each system table is associated with a data dictionary file. *See also* data dictionary and data dictionary file.

system transaction

A bundle of successful non-transactional operations and/or exclusive or concurrent transactions that the MicroKernel uses to gain better performance and to aid data recovery. Unless a system failure occurs, system transactions are transparent to clients. Unlike user transactions (exclusive or concurrent), users have no control over system transactions; users cannot abort system transactions.

Note: Do not confuse system transactions with exclusive or concurrent transactions. “System transaction” is a term specific to data recovery. Throughout this manual, the term “transaction” refers to an exclusive or concurrent transaction (also known as a user transaction).

table

A collection of data formatted in rows, where each row consists of column values. Vertical collections of column values in the table form columns, and each column contains one type of information, such as salary or revenue. Each row contains the same number of items and type of information as every other row in the table. The two types of tables are base tables and views. *See also* base table, column, column value, row, and view.

table definition

A table description stored in a data dictionary. *See also* data dictionary.

task

An instance of an application. *See also* application.

Transmission Control Protocol (TCP)

A connection-oriented communication protocol that uses the Internet Protocol (IP) for packet delivery.

temporary mask

An edit mask that remains in effect until you define a new one for the column with XQLFormat, compile a different SQL statement with the same cursor ID, or release the cursor ID. You create a temporary mask by placing the mask in square brackets directly following a column name in a SELECT statement or by using the XQLFormat function. *See also* default mask, edit mask, and temporary view.

temporary sort file

A temporary file that Scalable SQL builds if you sort a view by a column that is not defined as an index.

temporary file

A transient data file that improves performance for the duration of a task.

temporary view

A view that you do not name or store in the data dictionary. You create a temporary view using a SELECT statement, and you define a stored view using a SELECT clause within a CREATE VIEW statement. *See also* view.

thread

A separate unit of execution within a program. *See also* multithreading.

thunking

Thunking is the transition from executing 32-bit code to executing 16-bit code, or from executing 16-bit code to executing 32-bit code. The piece of code that makes the transition is called a thunk. This thunk is a mapping layer that converts data to allow 32-bit code and 16-bit code to communicate with each other. A thunk allows applications to use controls of another memory model without the application knowing the memory models are different.

transaction

A group of related operations that must be performed together and in a particular order to ensure logical data integrity.

transaction atomicity

The assurance that the MicroKernel completes all operations of a user transaction. If the MicroKernel cannot complete all operations, the MicroKernel rolls back all the operations so that the data files are in the same state as before the transaction began.

transaction control

A type of concurrency control that instructs the MicroKernel to lock a file (or the modified pages of a file) that an application opens or accesses within a Btrieve transaction. The file remains locked until the transaction ends or aborts. Other applications can open and read from the file while it is locked but cannot perform write operations or transactions on the file.

transaction durability

The assurance that the MicroKernel finishes writing to the transaction log when a client ends a transaction and before the MicroKernel returns a successful status code to the client. Because a user transaction is not necessarily the last element in a system transaction, following a system failure, the user transaction could be rolled back, even though the MicroKernel returns Status Code 0 to the client for the End Transaction operation. The MicroKernel does not guarantee transaction durability on files opened in Accelerated mode.

transaction log

One or more physical files, called log segments, that the MicroKernel uses to ensure transaction durability.

Transaction Tracking System (TTS)

A NetWare fault tolerance system that protects files from corruption by backing out incomplete transactions that result from a failure in a network component. Pre-6.0 Btrieve uses TTS to protect data files on a NetWare server. TTS must be active at the server, and the data files must be flagged as transactional.

Btrieve for NetWare 6.0 and later uses TTS somewhat differently. Applications can flag a file as transactional when they want to signal the MicroKernel that the file's integrity *must* be guaranteed. However, the MicroKernel does not actually use TTS to ensure integrity because TTS is not as fast as other means when used in conjunction with concurrent transactions. Instead, when the MicroKernel notices the TTS flag, it forces pages to be written chronologically (in order to ensure that Btrieve recovery mechanisms work properly).

Workstation engines disregard the TTS flag.

translation

Mapping characters in one character set to those in another. Translation is necessary for the database engine to support string types other than the engine’s native character set.

triggers

Dictionary objects that define a specific action for the DBMS to take whenever modifications to a particular data file take place. For example, you can define a trigger to call a stored procedure when an inventory quantity drops below a specified minimum.

TTS

See also transaction control and Transaction Tracking System (TTS).

tuple

See row.

unbound database

A database in which either the dictionary files are not bound or only some files are bound. An unbound database can be named or unnamed. See also bound database and bound file.

uniqueness

An index attribute that determines whether multiple rows can have the same index column value.

A key attribute that determines whether multiple records can have the same key value.

See also index attribute.

See also key attribute.

Update right

A Scalable SQL security right that allows users or user groups to update information in specified columns or tables.

update rule

A referential constraint that specifies how to apply updates to dependent tables. The update rule is always *restrict* , which prevents the MicroKernel from updating a row in a dependent table if the foreign key value for that row does not have an equivalent primary key value in the applicable parent table.

user

A person authorized to log in to a database when security is installed. If security is not installed, a user is any person that logs in to the database.

user group

Multiple users defined as a set. All the users in a group have the same security rights.

validation

A column attribute that specifies criteria for the acceptability of data that you add to a database. Scalable SQL can ensure that the data you enter is between specified minimum and maximum values (range checking), that it consists of specified characters (character validation), or that it matches a specified set of values (value validation). *See also* column attribute.

value

See column value.

value validation

See validation.

variable-length data type

A data type with a stored length that can vary from row to row within a table. *See also* fixed-length data type.

variablelength record

A record in a data file that contains a variablelength portion and a fixed-length portion. The fixed-length portions must be the same size in all the records in a given file, but the variablelength portions may vary in size. Consequently, the overall lengths of variablelength records may vary. *See also* fixed-length record.

variable page

See page.

Variable-Tail Allocation Table (VAT)

An array of pointers to the variable-length portion of Btrieve records. The VAT is implemented as a linked list.

Verify

A file open mode that causes the operating system to verify all write operations to a file. *See also* Accelerated, Exclusive, file open mode, Normal, and Read-Only.

view

A selection of rows and columns from one or more tables. A view can be treated as if all the data exists in a single table. The columns in a view can be from the same table or from different tables that have been joined together. You can create a view for a single use, or you can name and store it in the data dictionary for later retrieval. *See also* join and table.

virtual column

See computed column.

virtual table

A view or relation whose rows and columns do not have a one-to-one correspondence with base rows and columns, due to a join, aggregate, computed field, etc.

wait lock

A type of explicit record lock in which Scalable SQL and Btrieve do not return control to your task until they have obtained the lock on the record you requested.

In Scalable SQL, you request a wait lock by specifying a lock bias value on an XQLFetch call.

In Btrieve, you request a wait lock by specifying a lock bias value in operation calls.

See also explicit record locks and no-wait lock.

workstation MicroKernel

A version of the MicroKernel that runs as a standalone engine. All processing is performed on the workstation, and access to all files is through operating system calls. The operating system calls are either executed locally or redirected to the server. *See also* Btrieve and server-based MicroKernel.