R:Scope 11

Help Manual

R:Scope 11

Manual

by R:BASE Technologies, Inc.

Welcome to R:Scope 11!

R:Scope is an invaluable database repair tool that allows you to identify and repair broken R:BASE databases. As a separate program, R:Scope lets you examine and modify R:BASE database files outside R:BASE. R:Scope can check a database for problems and lets you look in the database files to diagnose a problem. Once you have determined the problem, R:Scope allows you to fix the structure of a database and correct data file errors.

Table of Contents

Part I

1 Introduction

1.1 Introducing R:Scope 11

R:Scope is an invaluable database repair tool that allows users to identify and repair broken R:BASE databases. As a separate program, R:Scope allows the examination and modification of R:BASE database files outside R:BASE. R:Scope can check a database for problems and look in the database files to diagnose possible problems. After doing so, R:Scope can fix the structure of a database and correct data file errors. R:Scope is also able to correct database file timestamps.

The R:Scope program includes all of the latest GUI features for fixing broken R:BASE databases. R:Scope include a database connection history option, View Data options, and additional printer setup output for the error log. An option is available to control the line numbers in the Check Screen window. Additional color options to enhance your environment in the Hex Viewer are also available. Multiple tables can be selected for the Check: Structure and Check: Data options. The table display from the Check and Fix menus have also been increased to better view your list of available tables. An option is also available to check indexes.

The Manual Fix option for the data file has an easy-to-use button tool bar and the ability to watch the current address while searching with display turned off. This is a great addition for developers with large databases. The ability to go directly to deleted rows is also available, which is helpful is restoring deleted records.

A status bar across the bottom of the window displays the computer, user, database, and current printer. The path of the current connected database appears in the R:Scope window caption so users know which database or database copy is connected. A Snapshot option has also been added to capture the current displayed screen.

The documentation includes chapters on understanding R:BASE databases, the basic repair strategy, and a list of error messages with suggestions.

R:Scope 11 is compatible with R:BASE 11 databases.

1.2 Copyrights

Trademarks

Printed: April 2025 in Murrysville, PA

First Edition

1.3 License

R:BASE TECHNOLOGIES, INC. LICENSE AGREEMENT

R:Scope 11 for Windows
Single Seat License

This is a legal agreement between you, the end user ("**Licensee**"), and R:BASE Technologies, Inc. ("**RBTI**"). Please read the terms and conditions of this License Agreement before using this software. By you selecting "I accept the license agreement" and clicking "Next" during product installation means you expressly accept the terms and conditions of this Agreement. If you do not accept the terms and conditions of this Agreement, you must stop installing the Product and click "Cancel". Your money will be refunded based upon proof of purchase, and in compliance with the return period described in the LIMITED WARRANTY below. A violation of the License, brings damage both financially and to the reputation of RBTI, and in the occurrence of either, both termination of the license agreement and civil damages will be vigorously sought. Once you have clicked "I accept the license agreement", you are entitled to use the Product under the following terms and conditions of this Agreement:

LICENSE
This RBTI License Agreement permits you to use one copy of the R:SCOPE computer software with associated utilities (the "**Program**") and accompanying user documentation (the "**Documentation**") on any single computer provided the Program is being used on only one computer at a time. A Program is "being used" on a computer when it is loaded into a temporary memory or installed on a hard drive in the computer. However, a copy of the Program installed on a network server for the sole purpose of distribution to other computers is not "being used". Each seat having access to the Program must have an appropriate license. If you anticipate that the number of seats with access to the Program will exceed the number of seats for which you are licensed, you must take steps to ensure that the appropriate licenses are obtained for each seat.

COPYRIGHT AND RESTRICTIONS
RBTI retains full ownership rights in the Program and Documentation. You may make a single copy of the Program; to be used solely for backup or archival purposes, or you may transfer the Program onto a single hard disk provided you keep the original solely for backup or archival purposes. Such copies shall be owned by RBTI. You may not copy any printed material or Documentation without prior permission. You may not decompile, disassemble cross-compile, reverse engineer, or make or distribute any other form of, or derivative work from, the Program. You may not obscure, alter or remove any RBTI copyright, trademark or proprietary rights notices.

TRANSFER OF LICENSE

You may not lend, rent or lease the Program or Documentation or any copies to any person. A transfer of license is only authorized through completion of the Consent to Assignment and Assumption of Contract agreement from RBTI. Upon license transfer, you are permanently giving such person possession of all copies of the Program and Documentation, are permanently giving up your right to use the Program and Documentation, and the recipient agrees to the terms of this License Agreement.

STEP UPS AND UPGRADE PURCHASES
As the Licensee, you are authorized to use the Program only if you are an authorized user of a qualifying product as determined by RBTI. The new license agreement takes the place of the agreement of the qualifying software you stepped up or upgraded from. After you upgrade, you may no longer use the software from which you upgraded. When you install the upgrade, you must uninstall the copy of the qualifying product.

TERM OF LICENSE
The licensing provided in this License Agreement is perpetual unless you violate any of its terms or conditions, at which time the license will automatically terminate. Upon termination you must return all copies of the Program and Documentation to RBTI or certify in writing to RBTI that all such copies have been destroyed and uninstalled from each workstation and/or network server. RBTI reserves the right to, at its expense and without prior notice, conduct periodic inspections for licensing compliancy. If licensee is found to be in violation of current agreement, RBTI may commence a civil action seeking fines, damages, attorney's fees and injunctive relief and may also, in appropriate circumstances, seek criminal prosecution.

SOFTWARE SUPPORT
The availability of software support services is subject to the End of Support (EOS) and End of Life (EOL) product life cycle, and to an active Software Assurance Plan. Where applicable, licensees will be provided with an option to upgrade to the current supported version of a software product.

LIMITED WARRANTY
RBTI warrants to you, as the initial user, the Program will perform substantially in accordance with the Documentation, provided it is used in unaltered form with functioning equipment and operating systems for which it was designed. RBTI will, at its option, with proof of payment within 30 days of the invoice date and after the Program with Documentation has been deactivated and uninstalled, and software installer, whether provided by download or other means, permanently deleted from all drives and folders, issue a full refund. These are your sole remedies for any breach of warranty. No exceptions will be made.

SERVICES PROVIDED WITH PURCHASE

1. 30-Day Limited Complimentary Technical Support

LICENSEE RESPONSIBILITIES

- To help us expedite the process and provide high quality assistance, the licensee must provide proof of purchase when calling. Proof of purchase is defined as the following: registration number, purchase date, version and build number, and company or individual to which product is registered.
- To have operating system, workstations, and local network installed and functional. RBTI will NOT be responsible for resolving issues not pertaining to the Program.
- Our support staff deals with advanced issues, therefore the person contacting RBTI for assistance should be the system administrator or have other R:BASE/SQL experience and be able to understand and implement the advice given.

R:BASE TECHNOLOGIES, INC. RESPONSIBILITIES
- To provide quality assistance in a timely manner to aid Licensee in the installation of the product within 30 days of the date of purchase.
- To provide a reasonable solution for any resolvable issue. Not all issues are resolvable, and therefore we will acknowledge the existence of known issues or "bugs" which we are presently aware of, that have no reasonable work-around.

RBTI reserves the right to limit the amount of support time allotted to a maximum of 2 HOURS during the 30-Day Complimentary Technical Support Period. We also reserve the right to limit the quantity of calls from a particular Licensee to 30 MINUTES in a single day. Issues are dealt with

on a case-by-case basis, and are handled at the discretion of the support agent assigned to the case. Complimentary Support is limited to INSTALLATION and ELEMENTARY CONVERSION related issues ONLY. Our support hours are from 10am. to 6pm. Eastern Time.

2. Fixes for Known Issues
RBTI will provide continued product fixes for known issues or "bugs" for 1 YEAR from the software purchase date. After 1 year, users are urged to acquire the necessary Software Assurance Plan to continue with product support and ongoing activations for reinstallations and license transfers.

3. Enhancement Requests
RBTI will provide continued product enhancements for requested features for 1 YEAR from the software purchase date. After 1 year, users are urged to acquire the necessary Software Assurance Plan to continue with product support and ongoing activations for reinstallations and license transfers.

NO OTHER WARRANTIES
Except as explicitly stated above, RBTI makes no express or implied warranties (including any warranties of merchantability or fitness) with respect to the character, function, or capabilities of the program, the documentation or their appropriateness for any user's purposes. RBTI cannot customize product(s) to meet specific needs in all cases. Examples of customization include, but are not limited to: special character sets, foreign language adaptations, specific device drivers or other localization issues. Under no circumstances will RBTI be held responsible for product functionality once alterations have been made to accommodate individual needs.

DISCLAIMER OF WARRANTY
The Program and the accompanying files are sold "as is" and without warranties as to performance or merchantability or any other warranties whether expressed or implied. Because of the various hardware and software environments into which the Program may be put, No warranty of fitness for a particular purpose is offered. Good data processing procedure dictates that any program be thoroughly tested with non-critical data before relying on it. The user must assume the entire risk of using the Program. Any liability of the seller will be limited exclusively to product replacement or refund of purchase price.

HIGH RISK ACTIVITIES
The Program is not fault-tolerant and is not designed, manufactured or intended for use or resale as on-line control equipment in hazardous environments requiring fail-safe performance, such as in the operation of nuclear facilities, air traffic control, aircraft navigation or communication systems, direct life support machines, or weapons systems, in which the failure of the Program could lead directly to death, personal injury, or severe physical or environmental damage ("High Risk Activities"). RBTI specifically disclaims any expression or implied warranty of fitness for High Risk Activities.

LIMITATIONS ON LIABILITIES
RBTI will not be responsible for any costs or damages associated with loss of the use of the Program or any other resources, loss of business or profits, any loss of data, any third-party claims or costs of substitute programs. In no event will RBTI be liable for any incidental, indirect, special, consequential or punitive damages suffered by the user or any other person or entity, whether from the use of the program or documentation, any failure thereof, or otherwise, even if RBTI or its dealers or agents are aware of the possibility of such damages. In no event will RBTI aggregate liability to you or anyone else exceed two times the license fee you paid for the program and documentation in this package. Because some states do not allow the limitation or exclusion of implied warranties and liabilities for consequential or incidental damages, the above limitations may not apply to you.

GENERAL
This License Agreement constitutes the full and complete agreement between parties. RBTI retains all rights not specifically granted herein. RBTI shall not be deemed to have waived any of its rights hereunder or under all copyright laws, trade secrecy laws or otherwise. This Agreement is intended as a legally binding agreement which will be enforced to the full extent permitted under applicable law, in whole or in part. If any one provision of this Agreement is declared invalid or unenforceable, all remaining provisions shall never less remain in effect. The laws of the state of Pennsylvania shall govern this Agreement. RBTI reserves the right to amend, alter, or revoke this agreement at any time. All revisions to this license agreement are available for inspection upon request, supersede conditions in all past agreements, and render prior license agreements void and unenforceable. Both parties named in this license will only be bound to the terms of the most current revision of this agreement.

U.S. GOVERNMENT RESTRICTED RIGHTS

Use, duplication or disclosure by the Government, its agents or employees is subject to all restrictions imposed by law, regulation or government directive, including but not limited to those restrictions set forth in DFARS 252.227-7013 and 48CFR 52.227-19, as applicable.

R:BASE Technologies, Inc.
https://www.rbase.com
rbaseinfo@rbase.com

Revised Monday, April 28, 2025

1.4 Complimentary Support

30 DAY LIMITED COMPLIMENTARY TECHNICAL SUPPORT

A. LICENSEE RESPONSIBILITIES.

1. To help us expedite the process and provide high quality assistance, the licensee must provide proof of purchase. Proof of purchase is defined as the following: registration number, purchase date, version and build number, and company or individual to which product is registered.

2. To have operating system, workstations, and local network installed and functional. R:BASE Technologies will NOT be responsible for resolving issues not pertaining to the software product.

3. Our support staff deals with advanced issues, therefore the person contacting R:BASE Technologies for assistance should be the system administrator or have other R:BASE/SQL experience and be able to understand and implement the advice given.

4. To have the database, application, and command files being reviewed, safely backed-up before attempting assistance. R:BASE Technologies will NOT be held responsible for lost data or corruption as a result of advice given.

B. R:BASE TECHNOLOGIES, INC. RESPONSIBILITIES.

1. To provide quality assistance in a timely manner to aid in the installation of the product and elementary conversion of database, application, and command files within 30 days of the date of purchase.

2. To provide a reasonable solution for any solvable issue. Not all issues may be solved, and therefore we will acknowledge the existence of known issues, or bugs, which we are presently aware of, that have no reasonable work-around.

R:BASE Technologies reserves the right to limit the amount of support time allotted to a maximum of 2 HOURS during the 30-Day Complimentary Technical Support period. We also reserve the right to limit the quantity of calls from a particular licensee to 30 MINUTES in a single day. Issues are dealt with on a case-by-case basis, and are handled at the discretion of the support agent assigned to the case. Complimentary Support is limited to INSTALLATION and ELEMENTARY CONVERSION related issues ONLY. Our support hours are Monday through Friday, from 10:00 AM to 6:00 PM (EST).

For application, design, or advanced conversion assistance, R:BASE Technologies offers Technical Support Plans of various types to meet your needs. Please visit the Support page at https://www.rbase.com/support for details and pricing.

Part II

2 Installation

2.1 System Requirements

The following system specifications are recommended for the optimal use of R:BASE and R:BASE-related software.

Workstation Hardware

- 2-Core 2GHz+ CPU
- 2 GB of available RAM (4 GB recommended)
- 2 GB of available hard disk space
- 1024x768 or higher resolution video adapter and display
- Standard mouse or compatible pointing device
- Standard keyboard

Server Hardware

- 2-Core 2GHz+ CPU
- 6 GB of available RAM (8 GB recommended)

Operating System

- Microsoft Windows 11 (Professional)
- Microsoft Windows 10 (Professional)
- Microsoft Windows Server 2025
- Microsoft Windows Server 2022
- Microsoft Windows Server 2019
- Microsoft Windows Server 2016

Network

- Ethernet infrastructure (Gigabyte recommended)
- Internet connection recommended, but not required, for license activation, software updates, and support
- Anti-virus programs should exclude the R:BASE program, and any add-on product, executable and database files

2.2 Software Installation

The installation of R:Scope is fully automated and does not require user intervention for the initial setup.

Run the installer ".exe", provided by download, while physically sitting at the workstation to begin the installation process, and read the installer screens for licensing and other information as the program installs.

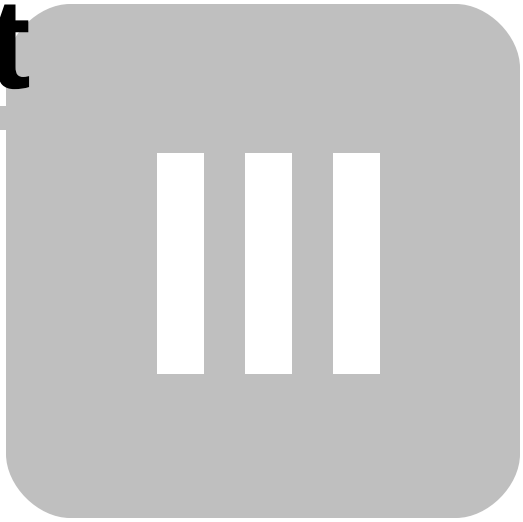
Installation Directory

C:\RBTI\RScope11

Files Installed

RScope11.exe
RBEngine11.dll
RScope11.chm
RScope11.pdf
License.rtf
ReadMe.txt

Part



3 Getting Started

R:Scope is a tool that lets you examine and modify R:BASE database files outside R:BASE. R:Scope can check a database for problems and lets you look in the database files to diagnose a problem. Once you have determined the problem, R:Scope allows you to fix the structure of a database and correct data file errors.

Once you are familiar with R:Scope, you can use R:Scope to check your database on a regular basis. You might want to make a habit of checking the database with R:Scope before packing the database with the R:BASE PACK or RELOAD commands, or before making a backup of the database files.

3.1 Backing Up Before Repairing

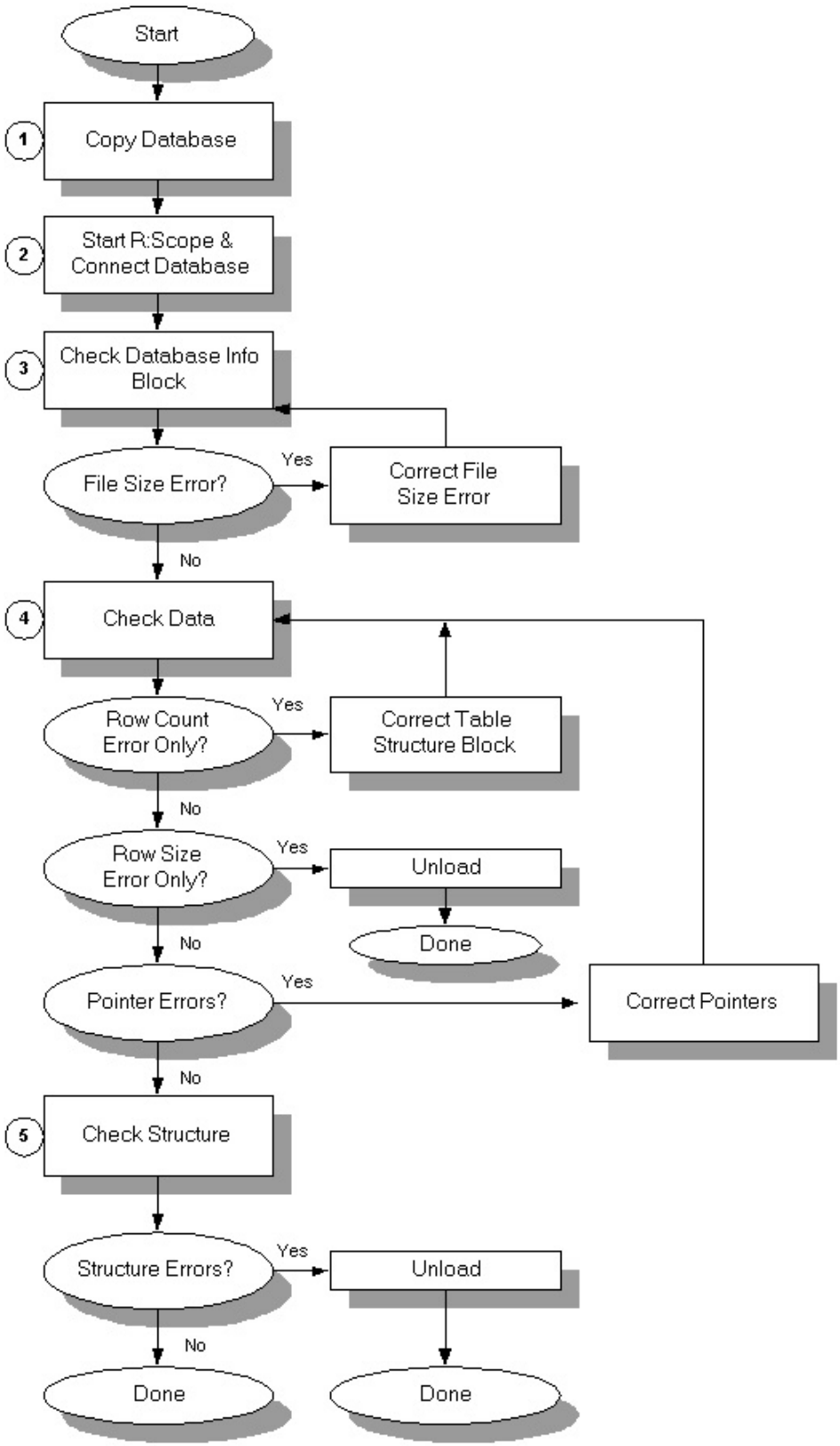
R:Scope was designed for use only when the database has a problem that cannot be resolved with R:BASE.

Before you work on a database with R:Scope, make a copy of the four database files. Copy the files using the operating system COPY command, or any program that makes backup copies. If you are unable to repair the database on the first attempt, having a copy of the files available will allow you to start again from the beginning.

Caution: Do not use the R:BASE BACKUP or UNLOAD commands to copy the database before repairing it. These commands back up only the data that R:BASE can find. If you use these commands on broken databases, you could lose data that otherwise might be recovered with R:Scope.

3.2 Basic Repair Strategy

Although it is not possible to list instructions that will fix every database on every occasion, you can use the following strategy to troubleshoot and repair most database problems. The following flow chart maps procedures for database repair. Refer to the text following the chart for an explanation of the procedures.



Copy the files using the operating system COPY command or any program that makes backup copies.

Start R:Scope and connect the database.

Use the R:Scope Check option to check the Database Info block in the structure file.

If R:Scope finds that the lengths of the data file or index file on disk do not match the lengths recorded in the structure file, use the R:Scope Fix option to correct these values in the structure file before continuing to check the database. Incorrect file lengths can cause R:Scope to report errors inaccurately when checking the data file.

Next, use the R:Scope Check option to check the data file for errors.

- If R:Scope finds row count errors only, this indicates a problem in the Table block of the structure file. Use the R:Scope Fix option to correct the Table block in the structure file. The row count for each table stored in the Table block must match the actual row count for each table in the data file.
- If R:Scope finds row size errors only, this indicates a problem in the structure file. Use the R:BASE UNLOAD and INPUT commands to recreate the database.
- If R:Scope finds pointer errors in addition to row count and row size errors or finds pointer errors only, this indicates a problem with row pointers in the data file. Use the R:Scope Fix option to correct the problems in the data file.

Use the R:Scope Check option to check the Table and Column blocks in the structure file. If R:Scope finds errors in the structure file, use the R:BASE UNLOAD and INPUT commands to recreate the database.

For more information about using R:Scope to check and fix the structure and data files, see Working with the Structure File, and Working with the Data File.

Part IV

4 Understanding R:BASE Databases

Before you attempt to fix a database with R:Scope, you must understand how an R:BASE database is structured. R:Scope is a powerful tool, but can damage a database if used incorrectly.

The Four Database Files

An R:BASE database consists of separate files stored on your hard disk. A file is simply a list of numbers, one after the other. The numbers are written in the file in binary form, that is, as bytes. The information in a typical R:BASE database can require a data file three megabytes long. That is, the file could contain three million bytes. The primary task of a database program such as R:BASE is to find the few bytes in the three million that correspond to a particular piece of information.

To perform this task, R:BASE creates files for each database. Databases created with R:BASE version 4.5++ and earlier have three files (.RB1, .RB2, and .RB3); R:BASE 5.x through X.5 (Version 10.5) create four files (.RB1, .RB2, .RB3, and .RB4). R:BASE Turbo V-8, eXtreme 9.x (64), X/X.5 Enterprise, and version 11 also creates four files, but with a different file extension (.RX1, .RX2, .RX3, and .RX4). When you name a database, R:BASE uses this name to create the files by appending a number and an extension to the name. For example, if you name a database *MyData,* R:BASE creates the following four files: MyData.RX1, MyData.RX2, MyData.RX3, and MyData.RX4.

Regardless of the version or R:BASE, the first three files are the same and serve the same purpose. The second file, MyData.RX2, contains the information stored in the database. The first and third files, MyData.RX1 and MyData.RX3, store the information that R:BASE uses to organize and find the information in MyData.RX2. The fourth file, MyData.RX4, stores binary or large text data, which is explained later.

You can think of a file as a long street of houses. Like a street that is divided into blocks of houses, a file can be divided into blocks of bytes. And, just as any house on a street can be identified by its address, any byte in a file can be found by its address. The address of a file location that is stored in a file is called a *pointer.* R:BASE uses pointers in all four of the database files to find its way to information in the database.

4.1 Database Structure in File 1

The structure of the entire database is stored in the RX1 file, for example, MyData.RX1. Throughout this help reference, this file is called the *structure file.* R:Scope options that allow you to see or change the structure of the database are working with the information in this file.

When you designed and built your database, you created a structure of tables and columns to hold information. When you open a database, R:BASE reads the structure file to find out how this structure is organized. The structure file tells R:BASE where and how the information in the data file is stored, where the indexes in the index file are located, and where and how the data is stored in the LOB file. R:BASE could neither find the data stored in the data file nor use the indexes stored in the index file without the information stored in the structure file. The information R:BASE learns from the structure file includes:

- The names of the tables and columns
- The number of tables and columns
- The order of tables and columns
- The number and size of rows in each table
- The settings for the database

Pointers link the database structure recorded in the structure file to specific, physical locations in File 2, the data file. For each table in the database, the structure file contains a pointer to the start of the table and a pointer to the end of the table. When you display data from an R:BASE table, R:BASE checks the pointers in the structure file to find the table's first row in the data file.

Once it has found the start of the table, R:BASE must determine where each column begins in each row that follows. In the structure file, R:BASE keeps information on the beginning location of each column for rows in the data file. The location is the number of bytes from the beginning of each row in the table. This information is stored in a two-byte unit called a *word.* For each column in the table, R:BASE finds its start in the data file by counting from the beginning of the row.

R:BASE updates the information in the structure file every time the database structure is modified, rows are added or deleted, the database settings are modified, the database is disconnected, or when you exit R:BASE. The update occurs as soon as the command is complete when CLEAR is set on.

4.1.1 Structure File Blocks

The structure file is divided into sections of bytes. Each section is called a *block* and contains a particular kind of data. R:Scope shows the starting point for each block, that is, the number of bytes from the beginning of the file. R:Scope also shows the length in bytes of each block.

Block Name	Contents
DB Info	Version flag; structure file block locations and lengths; data, index, and LOB file lengths; number of tables and columns. The Database Info option on the Display menu displays the information in this block.
Table	For each table, R:Scope shows table name, table #, row width, number of columns and rows, starting and ending data file pointers, and the number in the Column block for the first column in the table. The Table Structure option on the Display menu displays the information in this block.
Column	For each column, R:Scope shows the number in the Column block of each column, table name, column name, word offset, data type, column length, and index #. The Column Structure option on the Display menu displays the information in this block.
Index	For each index, R:Scope shows the index number, index name, table #, column name, index size, PK#, Next FK#, and pointer to File 3. The Index Structure option on the Display menu displays information in this block.

4.1.2 Database Version Flags

Version 11	Meaning
-1101	Normal mode version flag is set to -1101 when you disconnect a database or exit R:BASE normally.
-1102	Database is in read-only schema mode (STATICDB setting is on).
-1103	Database is in transaction processing (TRANSACT is set on).
-1104	Database is in transaction processing mode and is in need of recovery. Use the RECOVER command to restore the database.
-1105	Database is in transaction processing with compatibility (TRANSACT and COMPATIB is set on).
-1106	Database is in transaction processing mode with compatibility and is in need of recovery. Use the RECOVER command to restore the database.
-1111	Normal mode with FASTFK on.
-1112	Read-only schema (STATICDB on) with FASTFK.
-1113	Database is in transaction processing with FASTFK.
-1114	Database is in transaction processing mode with FASTFK and is in need of recovery. Use the RECOVER command to restore the database.

-1115	Database is in transaction processing with compatibility and fast FK (TRANSACT, COMPATIB, and FASTFK is set on).
-1116	Database is in transaction processing mode with compatibility and fast FK, and is in need of recovery. Use the RECOVER command to restore the database.
-1121	Normal mode with FASTLOCK on.
-1122	STATICDB on with FASTLOCK.
-1123	Database is in transaction processing with FASTLOCK.
-1124	Database is in transaction processing mode with FASTLOCK and is in need of recovery. Use the RECOVER command to restore the database.
-1125	Database is in transaction processing with compatibility and FASTLOCK (TRANSACT, COMPATIB, and FASTLOCK is set on).
-1126	Database is in transaction processing mode with compatibility and FASTLOCK, and is in need of recovery. Use the RECOVER command to restore the database.
-1131	Normal mode with FASTFK and FASTLOCK on.
-1132	STATICDB is on with FASTFK and FASTLOCK.
-1133	Database is in transaction processing with FASTFK and FASTLOCK.
-1134	Database is in transaction processing mode with FASTFK and FASTLOCK and is in need of recovery. Use the RECOVER command to restore the database.
-1135	Database is in transaction processing with compatibility, fast FK, and FASTLOCK (TRANSACT, COMPATIB, FASTFK, and FASTLOCK is set on).
-1136	Database is in transaction processing mode with compatibility, fast FK, and FASTLOCK, and is in need of recovery. Use the RECOVER command to restore the database.

4.2 Database Data in File 2

All the data in the database is stored in the RX2 file, for example, MyData.RX2. Throughout this help reference, this file is called the *data file.* R:Scope options that allow you to see or change rows of data are working with the information in this file.

The information stored in the database is recorded a row at a time in the data file. Although rows that belong to the same table can be physically next to one another in the file, they do not have to be. Each row in the file includes two pointers. R:BASE uses the pointers to find the rows in a table. One pointer shows the physical location for the start of the previous row in the table; the other shows the start of the next row. Each pointer is four bytes long. The pointers link the table's rows together in a long chain. Once R:BASE has found the table's first row by reading the structure file, it uses the first row's next row pointer to find the second row, the second row's next row pointer to find the third row, and so on through the table. If the pointers in a row become damaged, R:BASE will not be able to find all the rows in the table.

If a table contains LOB data types (VARBIT or VARCHAR), each row in the table with data in these columns contains pointers to the LOB data, which is stored in file 4 (RX4). For more information about LOB data, see The Database Large Object Data in File 4.

Each row of data also contains a two-byte value that stores the size of the row. The size of a row is the sum of the size of the columns making up the row. With the exception of tables that contain NOTE columns, all the rows in a particular table will be the same length. In the case of tables that include a NOTE column, the row length will vary depending on the size of the data in the NOTE column. The maximum size of a row is 32,768 bytes.

R:BASE updates the data file at the completion of every command that adds or deletes rows from a table when CLEAR is set on. R:BASE also updates the file when you disconnect a database or when you exit R:BASE. In the case of rows with NOTE data types, the data file is updated whenever data in a NOTE column is increased. Increasing the data stored in a NOTE type column causes the row to be moved depending on the NOTE_PAD setting.

4.3 Database Indexes in File 3

Values from indexed columns in the database are stored in the RX3 file, for example, MyData.RX3. Throughout this help reference, this file is called the *index file.*

When R:BASE builds an index for a column, it uses the column values as index values or creates index values based on the column values.

4.4 Database Large Object Data in File 4

All large binary objects (VARBIT data type) and large text objects (VARCHAR data type) are stored in the RX4 file, MyData.RX4. Throughout this help reference, this file is called the *LOB file,* for Large Object. LOBs include graphic files or large text data. R:BASE also stores forms, reports, and labels as LOBs in file 4.

The LOB data in file 4 contains a pointer back to a row in file 2. File 4 also contains a free block list. When data is added to File 4, the free block list is used to place the data within file 4. When a row with LOB data is deleted, the blocks in File 4 containing data associated with that row are added to the free block list and can be reused.

Part V

5 Using R:Scope

5.1 Tool Bar

Connect
Connects to a database and changes the drive or directory

Disconnect
Disconnects the current connected database

History
Provides a list of the previously connected databases

Snapshot
Captures the current screen shot and places the content into the Windows clipboard

Set
- Search - changes the type of search R:Scope performs when looking for rows in the data file
- Display - toggles the display of rows on and off when fixing the data file
- Contiguous - toggles the non-contiguous row warning on and off when fixing the data file
- Repeat - toggles R:Scope's command repeat feature when fixing the data file

Hex Viewer
Alter the color settings for the Hex Viewer

Line Count
Set the line count for the Check Screen window

Printer Setup
Set the current printer for output

Help
Displays the R:Scope in-line help documentation

About
Displays current information about the product such as version

Exit
Closes the R:Scope program

5.2 Display

The options on the Display menu show you information from the connected database's files.

- Database Info - displays information from the info block in the structure file
- Database Settings - displays the database settings
- Table Structure - displays information from the table block in the structure file
- Column Structure - displays information from the column block in the structure file
- Index Structure - displays information from the index block in the structure file

- Structure File - displays the contents of the structure file byte by byte
- Data File - displays the contents of the data file byte by byte
- Index File - displays the contents of the index file byte by byte
- LOB File - displays the contents of the LOB file byte by byte

- Chart Data File - matches the rows in the data file with their tables and displays the result, including the data file pointer values

- Chart LOB File - matches the rows in the data file to the corresponding LOB data in the LOB file. The non-matching LOB values, including the LOB pointer values, are displayed.

5.2.1 Database Info

When you open a database in R:BASE, R:BASE reads the start of the structure file first. This portion of the structure file gives R:BASE the most basic information about the database. With R:Scope, you can display this information, check it for accuracy, and edit it.

To display the opening information from the structure file, choose Display on the R:Scope Menu and choose Database Info. R:Scope displays the following information:

Version Flag
The version flag indicates the version and status of the database. The "(V11)" value specified that the database is specific to Version 11.

Owner Identifier
R:Scope shows the database owner identifier. If the database does not have an owner identifier, the identifier is shown as *public.*

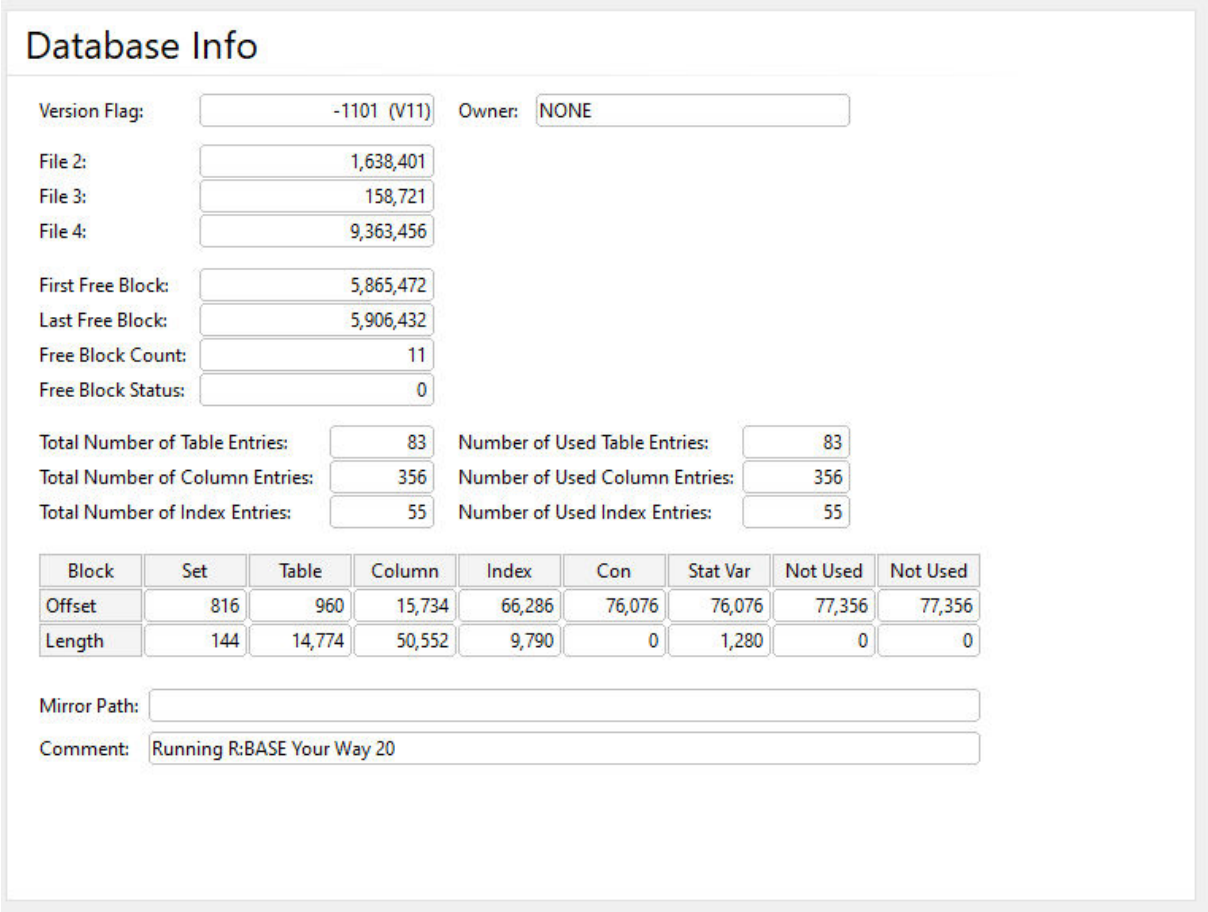
File Sizes
R:Scope shows the location where the next block will be added for the data file, index file, and the LOB file.

Free Block List
R:Scope shows the start and number of blocks for the free block list in the LOB file.

Number of Tables, Columns, and Indexes
R:Scope shows the number of tables, columns, and indexes in the database. Used and unused entries are counted. Unused entries can be cleared by using the RELOAD command to recover this space.

Mirror Path
The mirror path displays the location for the duplicate copy of the database.

Comment
R:Scope provides the comment for the connected database.



5.2.2 Database Settings

The structure file contains a block that stores settings for the database. Settings not included here can be changed only in R:BASE. R:Scope displays the settings as they are stored in the structure file. In some cases, this differs from the way the settings are displayed in R:BASE.

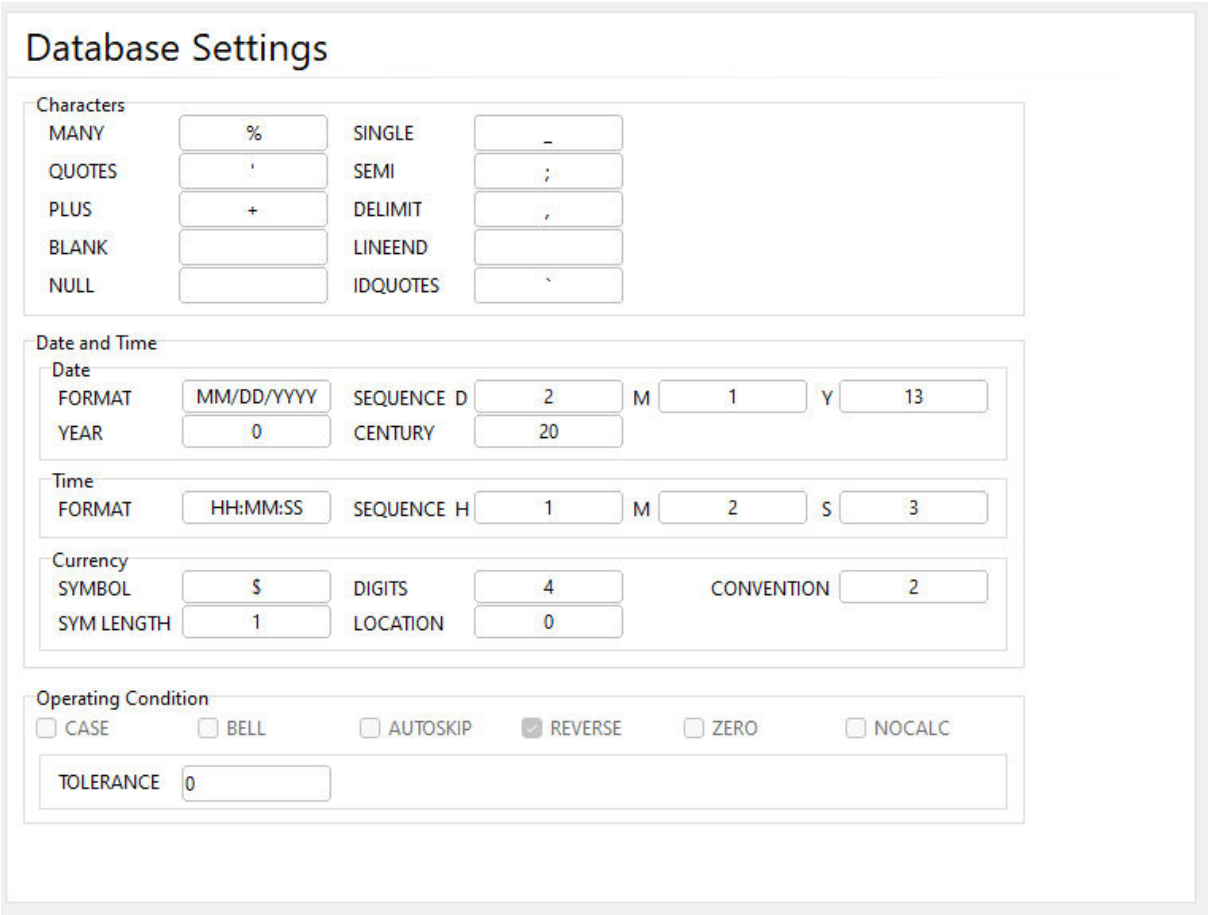
Characters
The settings are stored as they appear in R:BASE.

Date and Time
The date and time formats are stored as they appear in R:BASE. The date sequence is stored using 1, 2, and 3 to show the position in the sequence of the day, month, and year for dates, and of the hour, minute, and second for times. 0 indicates no position in the sequence.

Currency
The currency symbol and number of digits in the currency sub-units are stored as they appear in R:BASE. The format, shown in R:Scope in the Convention field, is stored as 1,2, or 3. A 1 corresponds to the R:BASE A format, 2 to the B format, and 3 to the C format. Formats other than B require you to change the delimiter to a character other than a comma. The position of the currency symbol before or after the value is stored as 0 or 1. Use 0 to indicate a prefix; use 1 to indicate a suffix. The symbol length designates the number of characters in the currency symbol.

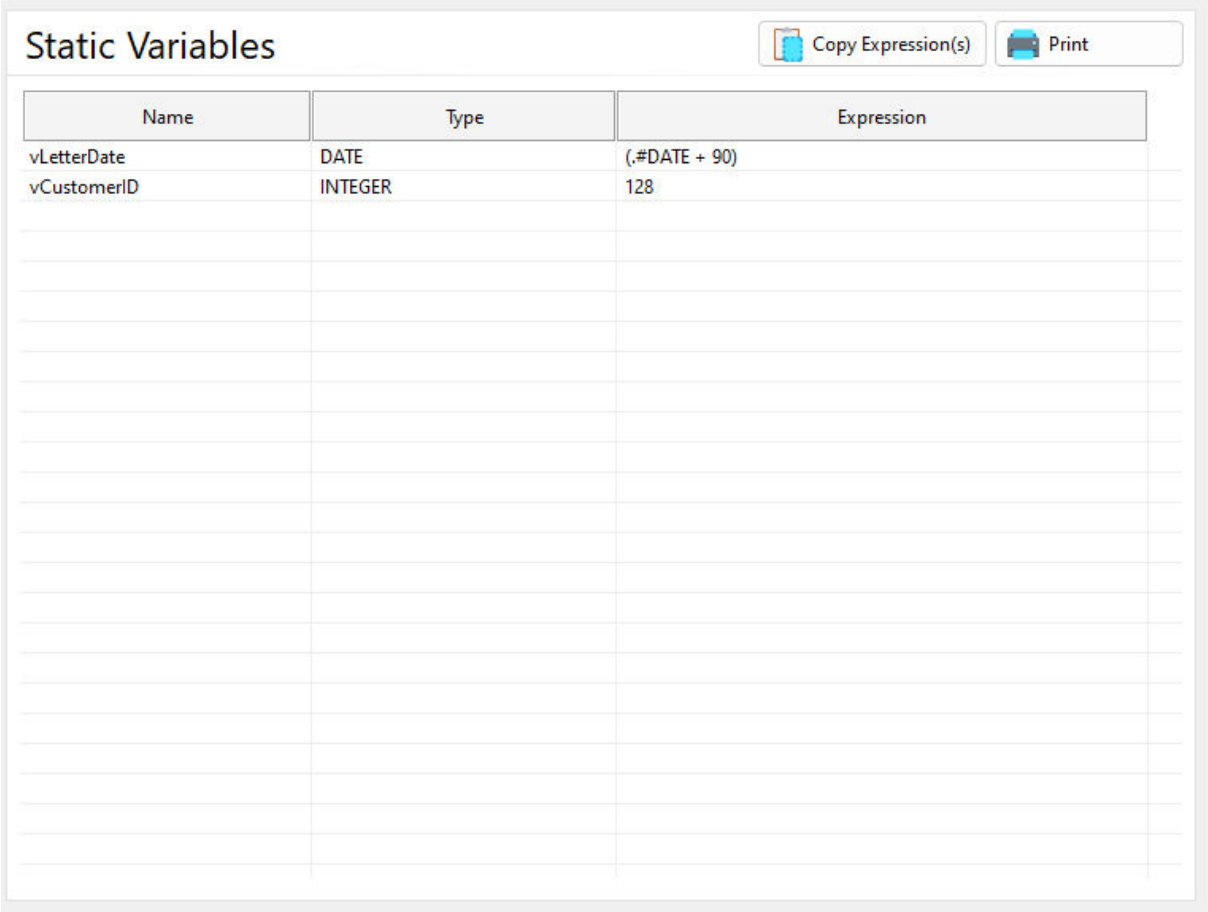
Operating Condition
The settings for CASE, BELL, AUTOSKIP, REVERSE, ZERO, and NOCALC are stored as they appear in R:BASE.

Tolerance
Tolerance is stored as it appears in R:BASE. The default value is 0.



5.2.3 Static Variables

To display the static variables, choose Display on the R:Scope menu and choose Static Variables. Static variables are variables which are part of a connected database. The Copy Expression(s) button allows for the selected static variables to be placed in the clipboard.



5.2.4 Table Structure

To display the Table block, choose Display on the R:Scope menu and choose Table Structure. This option displays a list of all tables in the open database, complete with structural information.

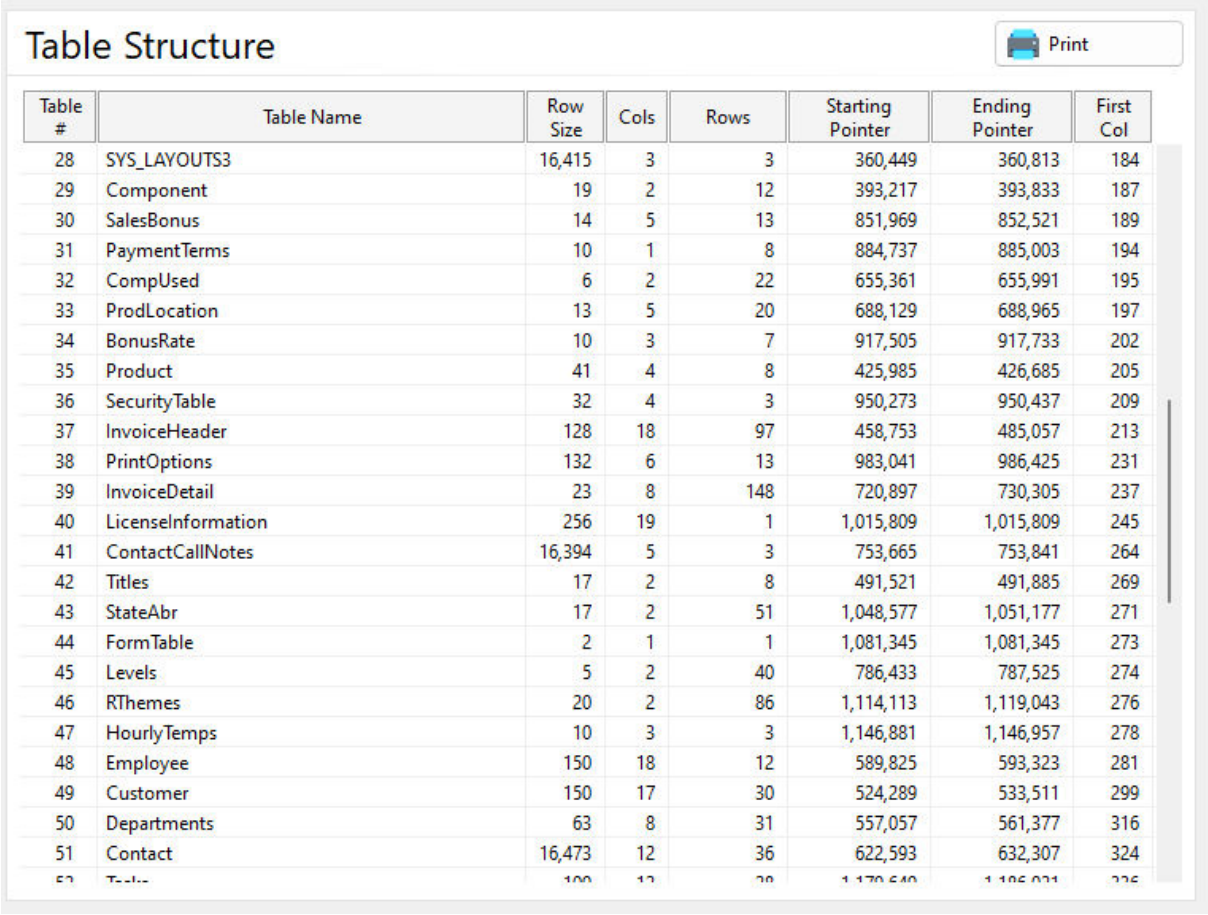


Table Structure

Print

Table #	Table Name	Row Size	Cols	Rows	Starting Pointer	Ending Pointer	First Col
28	SYS_LAYOUTS3	16,415	3	3	360,449	360,813	184
29	Component	19	2	12	393,217	393,833	187
30	SalesBonus	14	5	13	851,969	852,521	189
31	PaymentTerms	10	1	8	884,737	885,003	194
32	CompUsed	6	2	22	655,361	655,991	195
33	ProdLocation	13	5	20	688,129	688,965	197
34	BonusRate	10	3	7	917,505	917,733	202
35	Product	41	4	8	425,985	426,685	205
36	SecurityTable	32	4	3	950,273	950,437	209
37	InvoiceHeader	128	18	97	458,753	485,057	213
38	PrintOptions	132	6	13	983,041	986,425	231
39	InvoiceDetail	23	8	148	720,897	730,305	237
40	LicenseInformation	256	19	1	1,015,809	1,015,809	245
41	ContactCallNotes	16,394	5	3	753,665	753,841	264
42	Titles	17	2	8	491,521	491,885	269
43	StateAbr	17	2	51	1,048,577	1,051,177	271
44	FormTable	2	1	1	1,081,345	1,081,345	273
45	Levels	5	2	40	786,433	787,525	274
46	RThemes	20	2	86	1,114,113	1,119,043	276
47	HourlyTemps	10	3	3	1,146,881	1,146,957	278
48	Employee	150	18	12	589,825	593,323	281
49	Customer	150	17	30	524,289	533,511	299
50	Departments	63	8	31	557,057	561,377	316
51	Contact	16,473	12	36	622,593	632,307	324

Table Structure Block

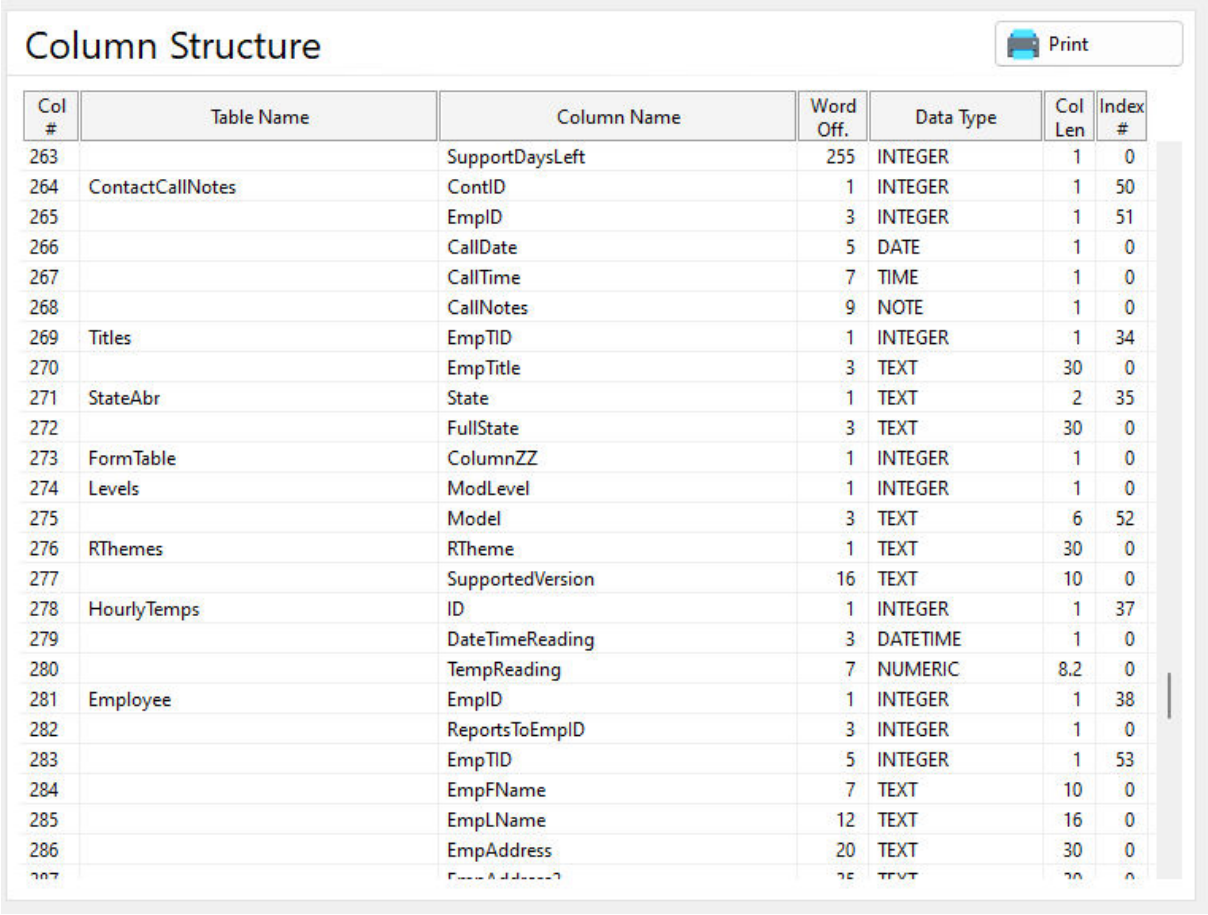
Structure Item	Description
#	The position number of the table in the database.
Table Name	The complete table name.
Row Size	The length of a single row of data in two-byte words. For tables containing NOTE columns, the row length is 16,384 words (32,768 bytes) plus the fixed row length.
Cols	The number of columns in the table.
Rows	The number of active rows in the table. A value of -1 indicates the table is a view. A value of -2 indicates it is a dBASE file. A value of -3 indicates it is an attached SERVER table. A value of -4 indicates it is a SYSTEM view.
Starting Pointer	The pointer value in bytes of the table's first row of data in the data file. This field will be zero for a table with no rows. This value links the structure file to the data file. R:BASE uses this value for a sequential search.
Ending Pointer	The pointer value in bytes of the table's last row of data in the data file. This field is zero for a table with no rows. This value links the structure file to the data file. R:BASE uses this value when adding or moving rows.
First Col	An integer showing where the table's first column is in the Column block's order of columns. This number links the Table block to the Column block within the structure file.

Column Structure Block

Structure Item	Description
#	A number showing the column's place within the Column block order. This number links the Column block to the Table block.
Table Name	The name of the table containing the column.
Column Name	The name of each column.
Word Offset	The value in words of the starting position of this column in a row of data. The fIrst column in a row always has an offset of 1. R:BASE uses the word offset to know where columns begin and end within rows retrieved from the data file.
Data Type	The data type of the column.
Column Length	The value is 1 for all columns except those with a NUMERIC or TEXT data type or an attached dBASE file or SERVER table. TEXT columns show the defined length of the column in bytes; NUMERIC columns show the precision and scale. For LONG VARCHAR and LONG VARBIT, the column length is 0. For VARCHAR and VARBIT, the length is the defined length of the column in bytes.
Index #	This value indicates the index information in the Index block. The value links the column with the Index block.

5.2.5 Column Structure

To display the column structure, choose Display on the R:Scope menu and choose Column Structure. This option displays a list of all columns in the current database, including all information for each column stored in the structure file.



Column Structure

Print

Col #	Table Name	Column Name	Word Off.	Data Type	Col Len	Index #
263		SupportDaysLeft	255	INTEGER	1	0
264	ContactCallNotes	ContID	1	INTEGER	1	50
265		EmpID	3	INTEGER	1	51
266		CallDate	5	DATE	1	0
267		CallTime	7	TIME	1	0
268		CallNotes	9	NOTE	1	0
269	Titles	EmpTID	1	INTEGER	1	34
270		EmpTitle	3	TEXT	30	0
271	StateAbr	State	1	TEXT	2	35
272		FullState	3	TEXT	30	0
273	FormTable	ColumnZZ	1	INTEGER	1	0
274	Levels	ModLevel	1	INTEGER	1	0
275		Model	3	TEXT	6	52
276	RThemes	RTheme	1	TEXT	30	0
277		SupportedVersion	16	TEXT	10	0
278	HourlyTemps	ID	1	INTEGER	1	37
279		DateTimeReading	3	DATETIME	1	0
280		TempReading	7	NUMERIC	8.2	0
281	Employee	EmpID	1	INTEGER	1	38
282		ReportsToEmpID	3	INTEGER	1	0
283		EmpTID	5	INTEGER	1	53
284		EmpFName	7	TEXT	10	0
285		EmpLName	12	TEXT	16	0
286		EmpAddress	20	TEXT	30	0

The table below shows the integers R:BASE uses to indicate data types and the column length for each data type in bytes. The word offset for each column is calculated by adding the column length in words to the offset of the previous column.

Column Data Type and Blocks

Data Type	Integer	Length in Bytes
INTEGER	1	4
REAL	2	4
TEXT	3	Defined length
DATE	4	4
TIME	5	4
CURRENCY	6	8
DOUBLE	7	8
NOTE	8	4
MUMERIC	9	8
DATETIME	10	8
VARCHAR	11	16
BIT	12	4 + defined length
BITNOTE	13	4
VARBIT	14	16
COMPUTED	-n	Length of resulting data type
BIGNUM	15	20
BSTR	16	0
GUID	17	16
BIGINT	18	1
SMALLINT	19	1

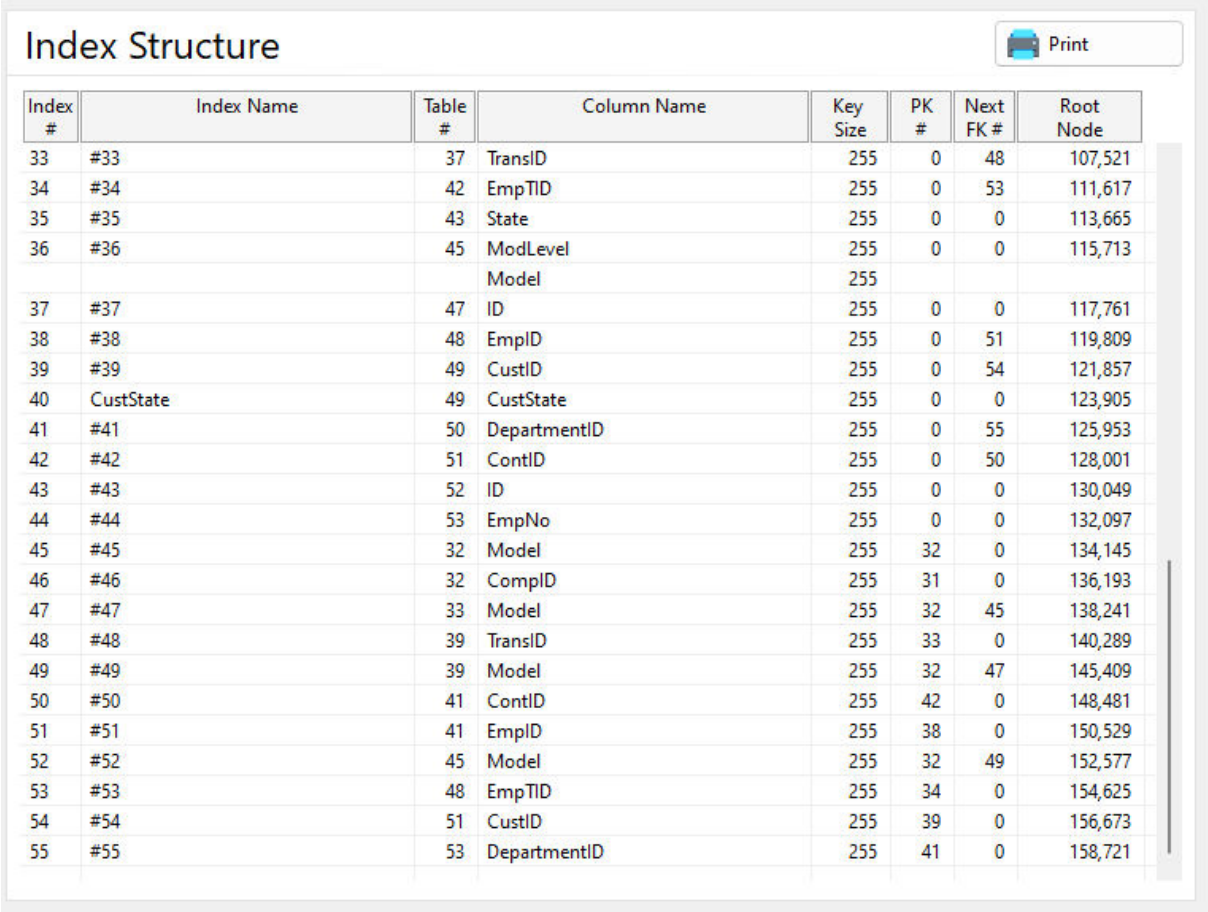
BOOLEAN	20	1
WIDETEXT	21	Defined length
WIDENOTE	22	Defined length

5.2.6 Index Structure

To display the Index block, choose Display on the R:Scope menu and choose Index Structure. This option displays a list of all indexes in the open database, complete with structural information.

Index Structure Block

Structure Item	Description
#	The position number of the index.
Index name	The index name
Table #	The table position number the index is associated with.
Column name	The name of the column the index is associated with.
Key size	The size of the index. 255 = non-text or full text. 0 = hashed. Other numbers are the number of characters used to preserve uniqueness.
PK#	Index # of the referenced primary key when the index is a foreign key.
Next FK #	Index # of the next foreign key when multiple foreign keys reference the same primary key.
Root Node	Pointer value to start of index information in the index file.



Index Structure

Print

Index #	Index Name	Table #	Column Name	Key Size	PK #	Next FK #	Root Node
33	#33	37	TransID	255	0	48	107,521
34	#34	42	EmpTID	255	0	53	111,617
35	#35	43	State	255	0	0	113,665
36	#36	45	ModLevel	255	0	0	115,713
			Model	255			
37	#37	47	ID	255	0	0	117,761
38	#38	48	EmpID	255	0	51	119,809
39	#39	49	CustID	255	0	54	121,857
40	CustState	49	CustState	255	0	0	123,905
41	#41	50	DepartmentID	255	0	55	125,953
42	#42	51	ContID	255	0	50	128,001
43	#43	52	ID	255	0	0	130,049
44	#44	53	EmpNo	255	0	0	132,097
45	#45	32	Model	255	32	0	134,145
46	#46	32	CompID	255	31	0	136,193
47	#47	33	Model	255	32	45	138,241
48	#48	39	TransID	255	33	0	140,289
49	#49	39	Model	255	32	47	145,409
50	#50	41	ContID	255	42	0	148,481
51	#51	41	EmpID	255	38	0	150,529
52	#52	45	Model	255	32	49	152,577
53	#53	48	EmpTID	255	34	0	154,625
54	#54	51	CustID	255	39	0	156,673
55	#55	53	DepartmentID	255	41	0	158,721

5.2.7 Structure File

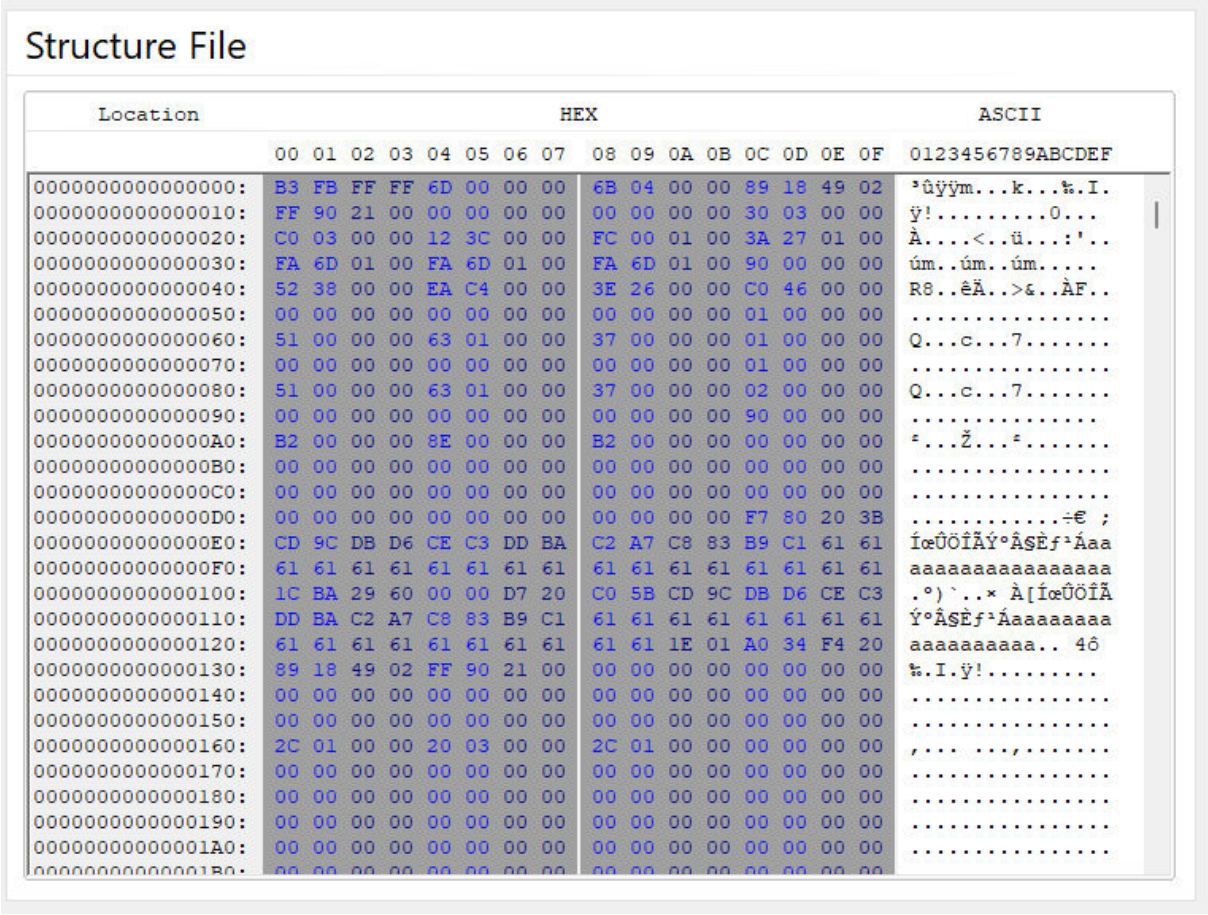
You can explore the contents of the whole structure file by choosing Display and choosing Structure File. R:Scope displays the contents of the structure file byte by byte. The display is organized in three columns: *Location, HEX,* and *ASCII.*

Location
R:Scope displays the file's data in lines of 8 bytes. The location column counts the display of bytes. Because the count starts at 0, when the location is 128, the first byte in the line is the 129th byte in the file.

HEX
The HEX column displays the hexadecimal value for each byte in the file. Each row shows 16 values.

ASCII
The ASCII column displays the ASCII value for each byte in the file. Each row shows 16 values.

You can scroll the file up and down on the screen. Press the [Up Arrow] or [Down Arrow] key to move a line at a time, and the [Page Up] and [Page Down] keys to move a screen at a time. To find out where the various blocks of information in the structure file begin, choose the Database Info option from the Display menu. For example, when you choose this option, R:Scope might show that the Table block begins 424 bytes into the file.



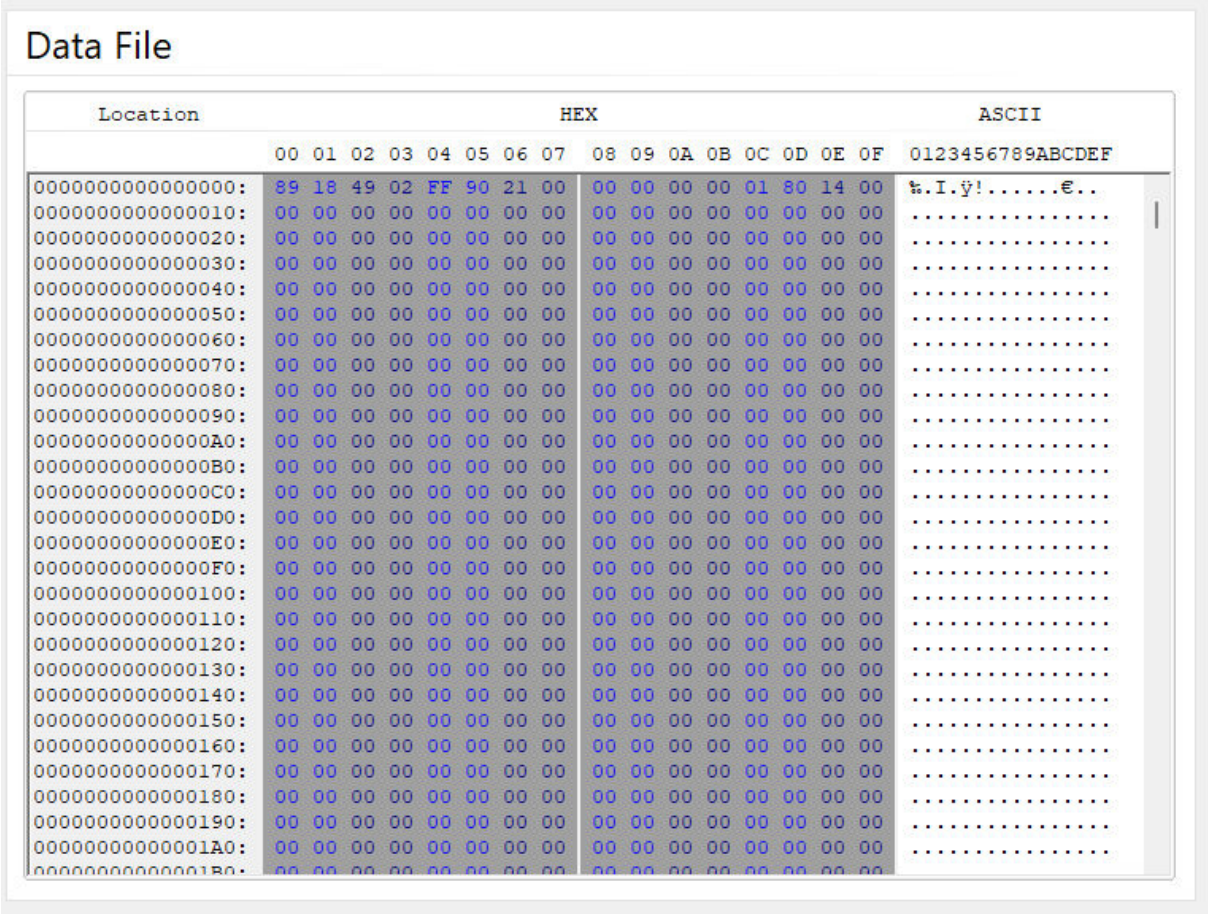
5.2.8 Data File

You can explore the Data File contents by choosing Display and choosing Data File. To look at rows from a particular table, choose Table Structure from the Display menu to find the starting pointer value for the first row in the table. R:Scope displays the contents of the structure file byte by byte and organizes the display in three columns: *Location, HEX,* and *ASCII*.

Location
R:Scope displays the file's data in lines of 8 bytes. The location column counts the display of bytes. Because the count starts at 0, when, for example, the location is 128, the first byte in the line is the 129th byte in the file. If you specified an address that is not a multiple of 16, the data for the address will begin somewhere in the first line R:Scope displays.

HEX
The HEX column displays the hexadecimal value for each byte in the file. Each row shows 16 values.

ASCII
The ASCII column displays the ASCII value for each byte in the file. Each row shows 16 values. When you are looking at data values, only data with a TEXT, NOTE, or NUMERIC data type will be comprehensible. You can scroll the file up and down with the [Down Arrow] and [Up Arrow] keys and [Page Up] and [Page Down] keys.



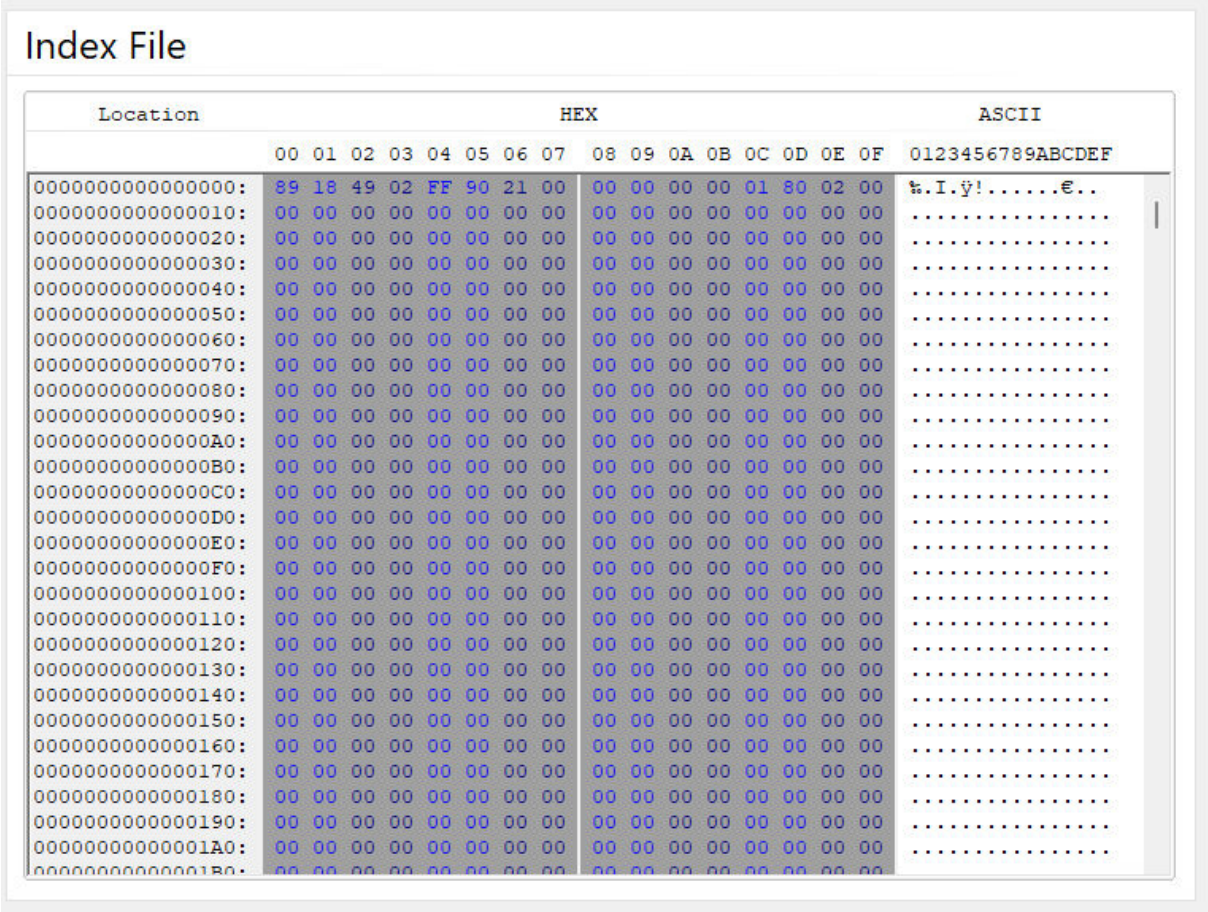
5.2.9 Index File

You can explore the Index File contents by choosing Display and choosing Index File. To look at rows from a particular table, choose Table Structure from the Display menu to find the starting pointer value for the first row in the table. R:Scope displays the contents of the structure file byte by byte and organizes the display in three columns: *Location, HEX,* and *ASCII*.

Location
R:Scope displays the file's data in lines of 8 bytes. The location column counts the display of bytes. Because the count starts at 0, when, for example, the location is 128, the first byte in the line is the 129th byte in the file. If you specified an address that is not a multiple of 16, the data for the address will begin somewhere in the first line R:Scope displays.

HEX
The HEX column displays the hexadecimal value for each byte in the file. Each row shows 16 values.

ASCII
The ASCII column displays the ASCII value for each byte in the file. Each row shows 16 values. When you are looking at data values, only data with a TEXT, NOTE, or NUMERIC data type will be comprehensible. You can scroll the file up and down with the [Down Arrow] and [Up Arrow] keys and [Page Up] and [Page Down] keys.



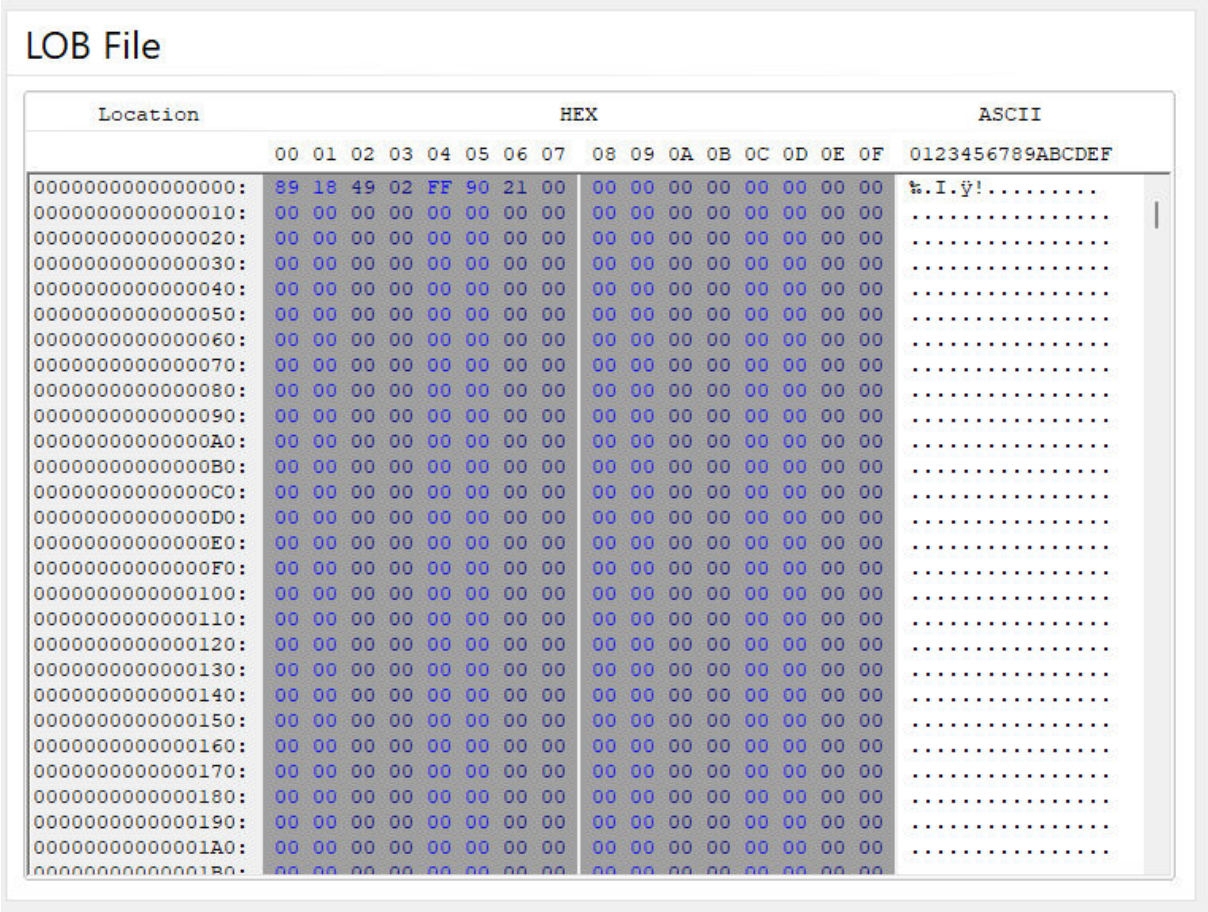
5.2.10 LOB File

You can explore the LOB File contents by choosing Display and choosing LOB File. To look at rows from a particular table, choose Table Structure from the Display menu to find the starting pointer value for the first row in the table. R:Scope displays the contents of the structure file byte by byte and organizes the display in three columns: *Location, HEX,* and *ASCII*.

Location
R:Scope displays the file's data in lines of 8 bytes. The location column counts the display of bytes. Because the count starts at 0, when, for example, the location is 128, the first byte in the line is the 129th byte in the file. If you specified an address that is not a multiple of 16, the data for the address will begin somewhere in the first line R:Scope displays.

HEX
The HEX column displays the hexadecimal value for each byte in the file. Each row shows 16 values.

ASCII
The ASCII column displays the ASCII value for each byte in the file. Each row shows 16 values. When you are looking at data values, only data with a TEXT, NOTE, or NUMERIC data type will be comprehensible. You can scroll the file up and down with the [Down Arrow] and [Up Arrow] keys and [Page Up] and [Page Down] keys.



5.2.11 Chart Data File

R:Scope can chart the distribution of rows in the data file for you. Rows belonging to a table might not be physically contiguous in the data file. In other words, the rows belonging to a particular table might be scattered in various locations in the data file. When you choose this option, R:Scope finds each contiguous group of rows in the data file and then matches each group with its table. R:Scope reports:

- The name of the table to which the group belongs. If R:Scope can't match a group of rows with a table, R:Scope reports the table as UNKNOWN.
- The beginning and ending pointer for each contiguous group of rows.
- The number of rows in the group.
- The minimum and maximum row length for the rows in the group. The lengths will be the same unless the rows include a NOTE column.

R:Scope lists the table for a group of rows that is not matched to a table as UNKNOWN. Such unknown groups can come about in the following ways:

- When you delete all rows from a table, the rows remain in the database but are no longer associated with the table.
- When you modify a table's structure in R:BASE, R:BASE changes the table's structure and creates a new set of rows for it based on the new structure. The old rows remain in the database but are no longer associated with a table.
- When you remove a table from the database, the table's rows remain in the database but are no longer associated with a table.

Use the Chart Data File option to find the data file pointer locations for a table's rows or to find pointer locations for rows that have been deleted from a table but still exist in the database.

To chart the data file, choose Display and choose Chart Data File. R:Scope analyzes the data file and then displays the results on the screen.

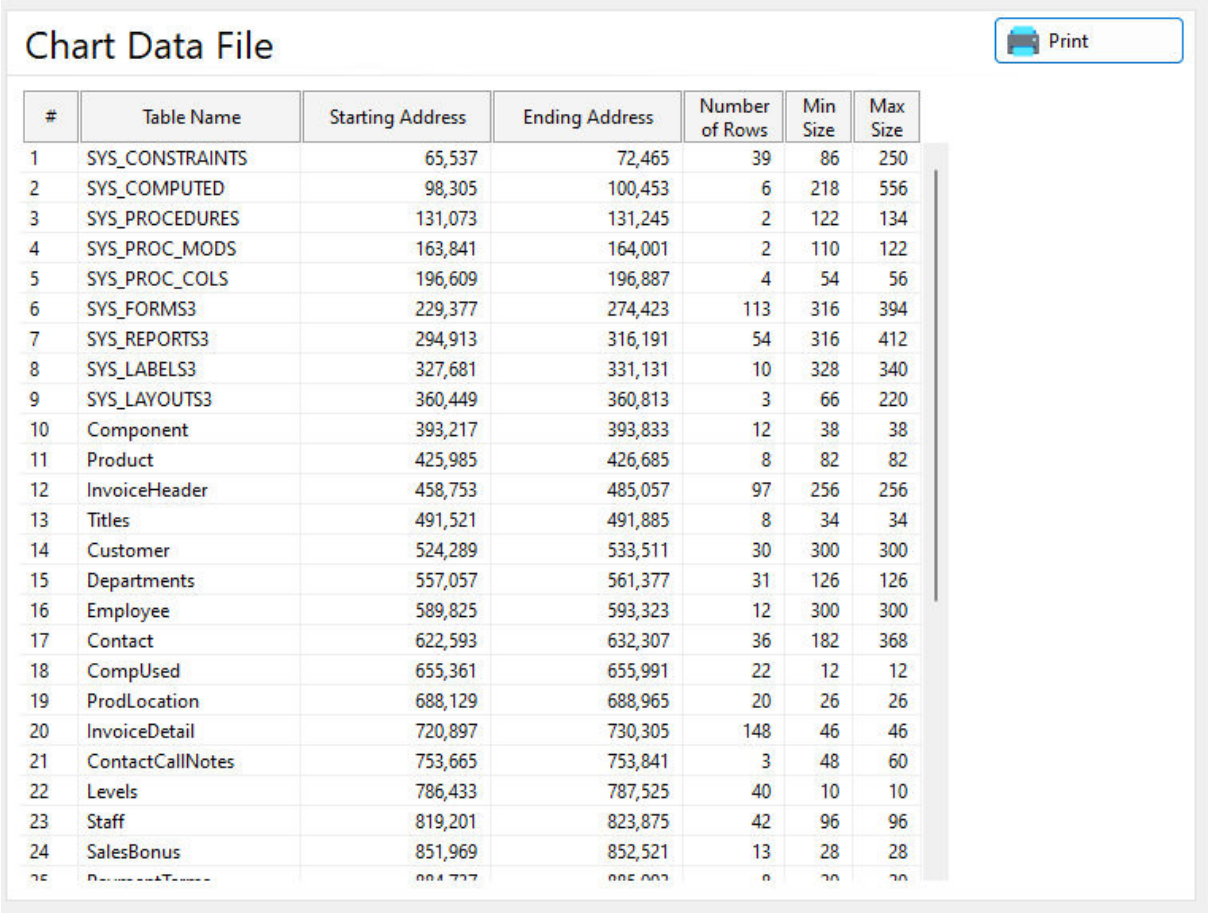


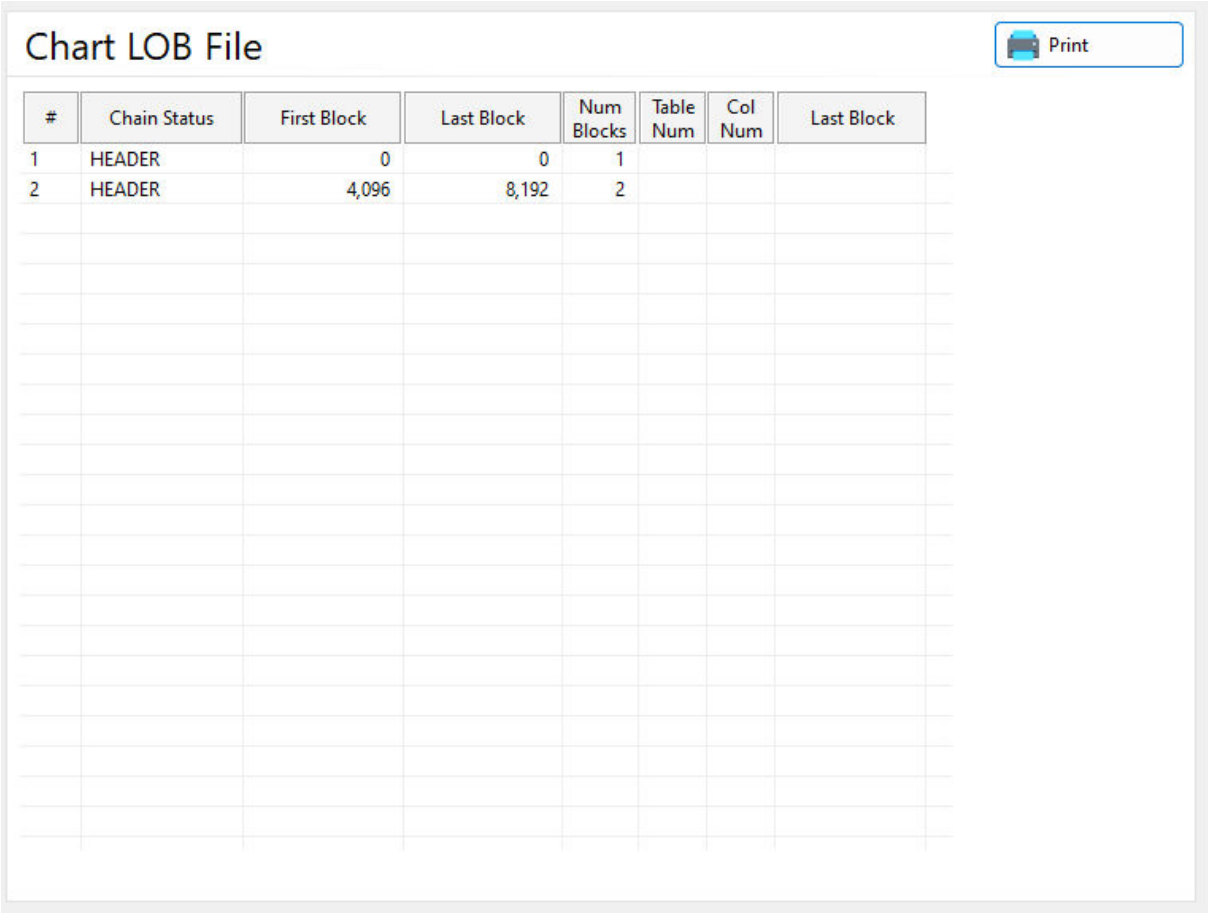
Chart Data File

Print

#	Table Name	Starting Address	Ending Address	Number of Rows	Min Size	Max Size
1	SYS_CONSTRAINTS	65,537	72,465	39	86	250
2	SYS_COMPUTED	98,305	100,453	6	218	556
3	SYS_PROCEDURES	131,073	131,245	2	122	134
4	SYS_PROC_MODS	163,841	164,001	2	110	122
5	SYS_PROC_COLS	196,609	196,887	4	54	56
6	SYS_FORMS3	229,377	274,423	113	316	394
7	SYS_REPORTS3	294,913	316,191	54	316	412
8	SYS_LABELS3	327,681	331,131	10	328	340
9	SYS_LAYOUTS3	360,449	360,813	3	66	220
10	Component	393,217	393,833	12	38	38
11	Product	425,985	426,685	8	82	82
12	InvoiceHeader	458,753	485,057	97	256	256
13	Titles	491,521	491,885	8	34	34
14	Customer	524,289	533,511	30	300	300
15	Departments	557,057	561,377	31	126	126
16	Employee	589,825	593,323	12	300	300
17	Contact	622,593	632,307	36	182	368
18	CompUsed	655,361	655,991	22	12	12
19	ProdLocation	688,129	688,965	20	26	26
20	InvoiceDetail	720,897	730,305	148	46	46
21	ContactCallNotes	753,665	753,841	3	48	60
22	Levels	786,433	787,525	40	10	10
23	Staff	819,201	823,875	42	96	96
24	SalesBonus	851,969	852,521	13	28	28

5.2.12 Chart LOB File

Use the Chart LOB File option to show chains in File 4 that are not owned by File 2 values.



5.3 Check

The options on the Check menu check the database for errors. When R:Scope checks the database, it writes the results to a file called ERROR.LOG and also displays them on the Check Screen, which can be printed. Each time you start R:Scope or connect a database, R:Scope erases the old ERROR.LOG file in the current directory.

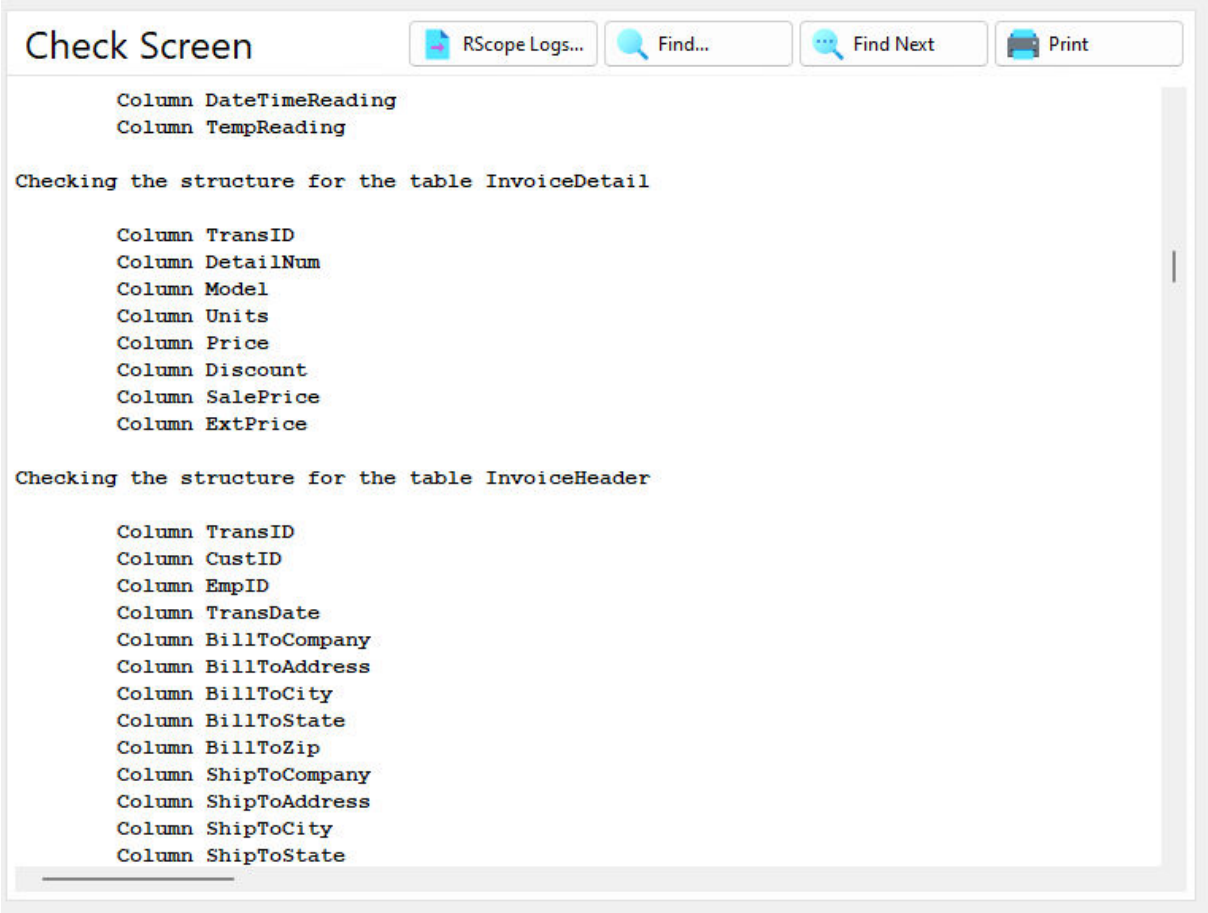
- Database Info - checks the Database Info block of the structure file for errors
- Structure - checks the Table and Column blocks of the structure file for errors
- Data - checks the data file and the LOB file for errors
- Indexes - checks indexes in the index (file 3) and walks through every pointer to the data file (file 2)
- Database Files Timestamps - checks for matching date/time stamps for the four database files
- Clear Check Screen - clears the Check Screen

Each time you use a Check menu option, R:Scope appends the results to the ERROR.LOG file and to the contents of the Check Screen. R:Scope displays the current results in bright text; previous results appear dimmed. R:Scope always places the cursor at the end of the most recent results. Display the last Check Screen result (without initiating any new results) with the Display Check Screen option from the Check menu. By collecting the results of the checks you perform on the database in a file, R:Scope allows you to refer to them again and to print them out. Whenever you start a new R:Scope session or connect to a database, R:Scope clears the Check Screen and creates a new ERROR.LOG file.

When you are working with the data file on the Fix: Data screen, display the Check Screen by selecting the "Check" tab, or press [Page Down] and return with [Esc]

Searching the Check Screen is possible using the "Find" and "Find Next" buttons.

You can print the Check Screen information with the Print button. R:Scope sends the contents of the ERROR.LOG file to your printer. Set the destination of the printer output with the Printer option on the Set menu where you can select a port to print to.



5.3.1 Database Info

R:Scope can verify that the information displayed in the Database Info dialog box is correct. To check the information, choose Check on the R:Scope menu and choose Database Info. R:Scope displays the information on the Check Screen.

R:Scope checks the following:

Version Flag
The version number must be a recognized version number.

Structure File Offsets and Lengths
The offset of a block must be greater than or equal to the offset of the previous block plus the length of the previous block.

Data File length
R:Scope checks the length stored in the structure file against the actual length of the file on disk as shown by the DIR command. The actual size must be within the range (File 2 size - 8912) to (File 2 size - 1). Next File 2 block is normally the actual size of (File 2 + 1).

Index File Length
R:Scope checks the length stored in the structure file against the actual length of the file on disk as shown by the DIR command. The actual size of the file must be within the range (File 3 size - 512) to (File 3 size + 4095).

LOB File Length
R:Scope checks the length stored in the structure file against the actual length of the file on disk as shown by the QIR command. The actual size of the file must be between (Next File 4) block and (Next File 4 block + 32,768).

LOB File Free Block List
R:Scope checks for free LOB blocks that can be reused.

Number of Tables
R:Scope checks that the number stored in the structure file is within the maximum range for an R:BASE database (1-32,000).

Number of Columns
R:Scope checks that the number stored in the structure file is within the maximum range for an R:BASE database (1-32,000).

Number of Indexes
R:Scope checks that the number stored in the structure file is within the maximum range for an R:BASE database (1-32,000).

Note that an error in the file length stored by the structure file can cause errors to appear in the data file even though the data file is correct. If R:Scope detects an error in the length of the structure file, you should correct it before checking the data file for errors.

5.3.2 Structure

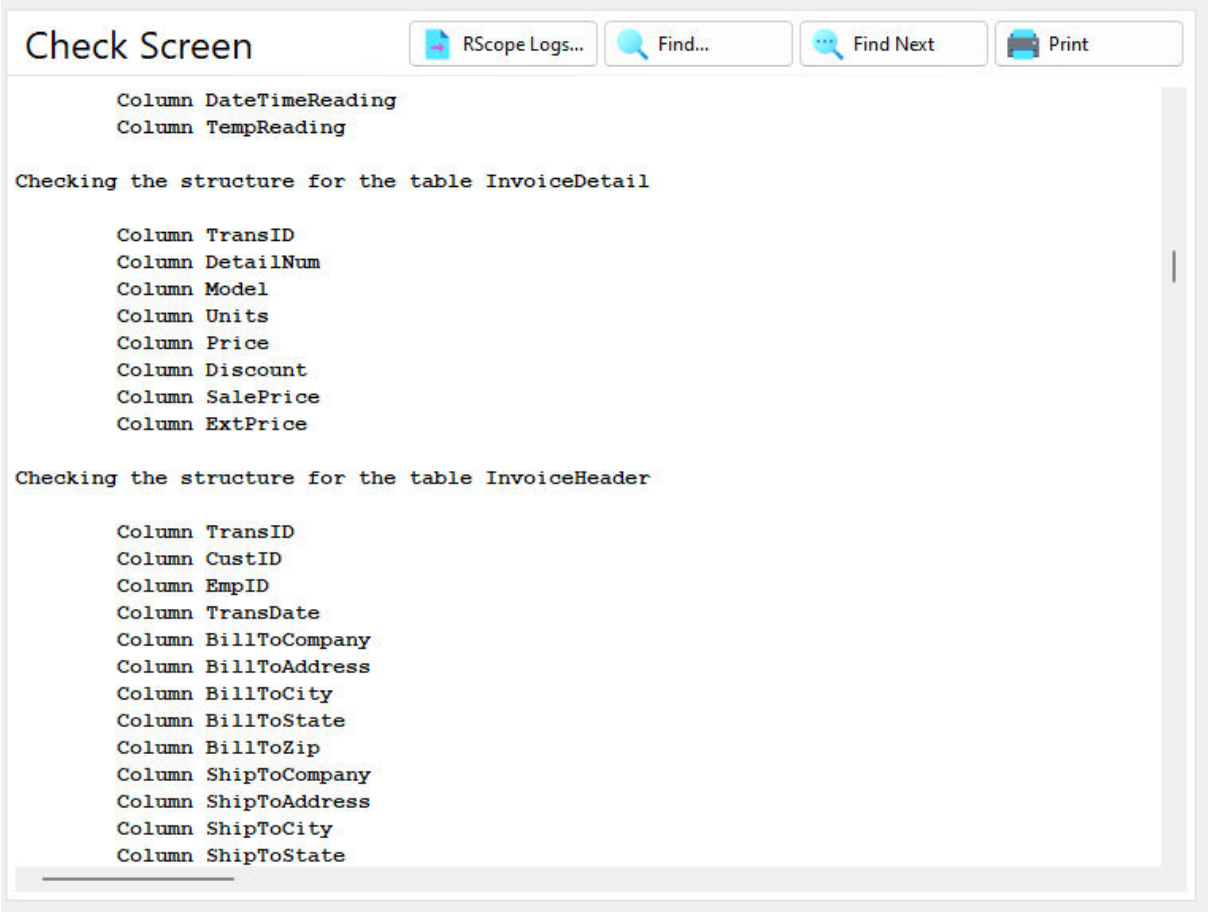
R:Scope will let you see, check, and change the information in the structure file. The structure file contains information about the whole database, the database tables, and the columns.

To check the information in both the Table and Column blocks:

1. Choose Check on the R:Scope menu and choose Structure. R:Scope displays a list of tables in the database.

2. With the mouse, select the table to check and select the "OK" button. To select more than one table, hold down the [Ctrl] key while selecting tables. To check all the tables, choose the "Select All Tables" button. The "Display Errors Only" check box is available to limit the Check Screen output to tables where errors are encountered.

3. R:Scope displays the information on the Check Screen.



For each table, R:Scope checks the following:

- Whether the table is a table (has a positive number of rows), a view, a dBASE file, an attached SERVER table, or a SYSTEM view.
- Each row width should be between 2 and 16,384 words.
- The number of columns for the table is no more than 2,048.
- The column number for the first column in the table is between 1 and the maximum value for the version of R:BASE being used.
- The starting pointer is between 0 and the current size of the data file. A value outside that range indicates a problem.
- The ending pointer is between the first row pointer and the size of the data file. A value out side that range indicates a problem.
- The total row length for all columns in a table must equal the value stored in the Table block of the structure file.

For each column, R:Scope checks the following:

- Data type validity.
- If a column is data typed TEXT, the length must be between 1 and 1500 bytes.
- The word offset for a table's first column should be 1. The word offset for each remaining column should equal the previous column's length in words plus the offset of the previous column.

5.3.3 Data

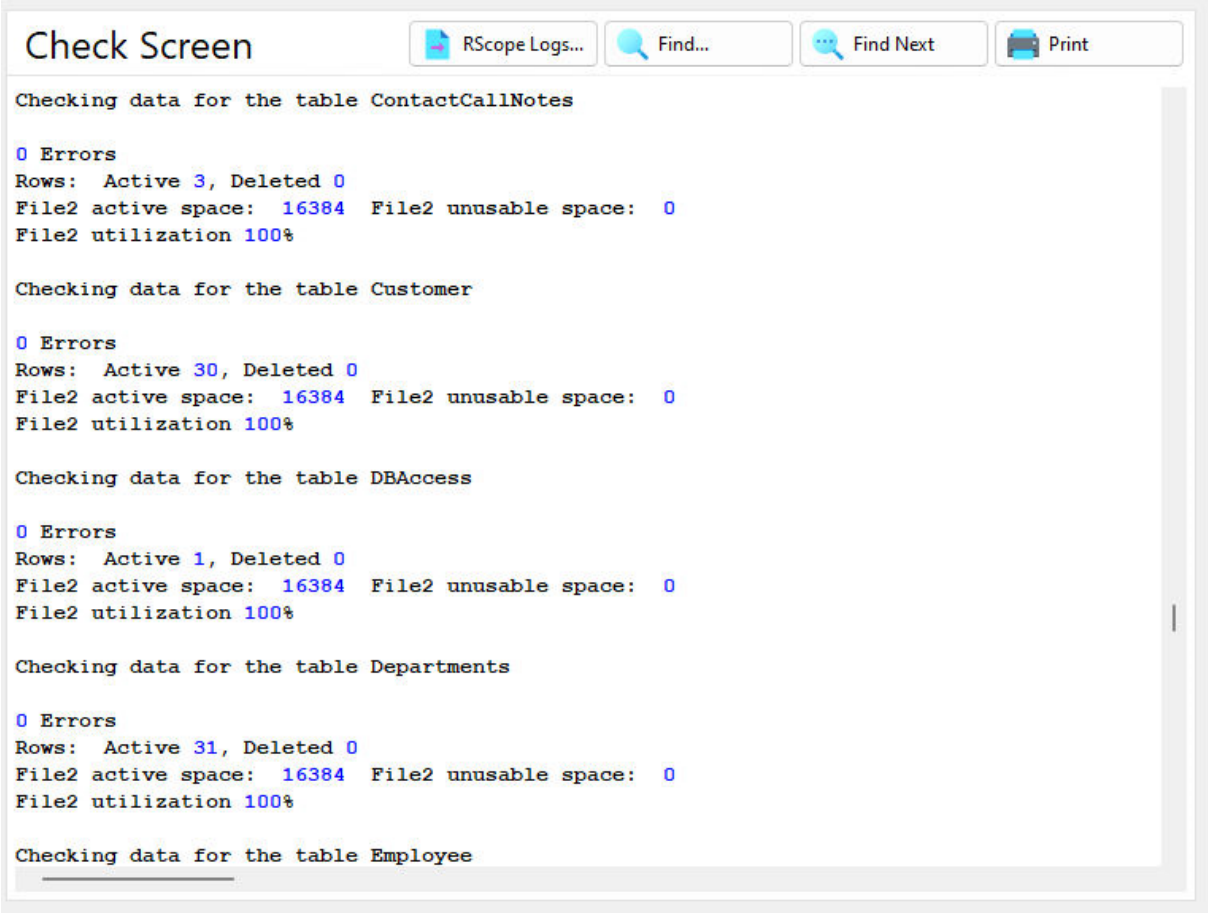
R:Scope checks the data file using the row pointers. R:Scope finds the first row for the table by referencing the starting pointer in the Table block of the structure file. Then R:Scope uses the next row pointers in the data file to chain through the rows for the table and ensures that the previous row pointer is valid.

To check the data file for errors:

1. Choose Check on the R:Scope menu and choose Data.

2. R:Scope displays a menu of tables. With the mouse, select the table to check and select the "OK" buton. To select more than one table, hold down the [Ctrl] key while selecting tables. To check the entire data file, choose the "Select All Tables" button. The "Display Errors Only" check box is available to limit the Check Screen output to tables where errors are encountered.

3. As it checks a table, R:Scope displays an expanding bar showing its progress through the table's rows. If R:Scope is unable to complete checking a table due to an error, the white bar indicates roughly how much of the table R:Scope checked. When R:Scope has finished checking, the results are displayed on the Check Screen.



R:Scope checks the following:

Row Size
For each row in the table, the size must match the value for the table's row width recorded in the Table block of the structure file. If the table contains NOTE columns, the row size must be in a valid range.

Row Count
The number of rows in the table must match the value for the table's number of rows recorded in the Table block of the structure file.

Last Row Pointer
The value of the pointer for the last row in the table must match the value for the ending pointer recorded in the Table block of the structure file.

Previous Row Pointer
For each row, the previous row pointer must be a value between the values for the table's starting pointer in the structure file's Table block and the current row address. The value should equal the address of the last row R:Scope encountered. The first row in the table has a previous row pointer value zero.

Next Row Pointer
For each row, the next row pointer must be a value between the values for the current row address and the table's ending pointer in the structure file's Table block. The last row in the table has a next row pointer value of zero.

New Location Pointer
When a row contains a NOTE column and the data in the NOTE column has increased, RBASE moves the row to the end of the table. In the first four bytes at the row's old location, R:BASE leaves a pointer to the row's new location and makes the row size 0. The pointer must be a value between the values for the current row address and for the ending pointer recorded in the Table block of the structure file.

LOB
For each row containing LOB data, the row must point to a valid row in File 4, this row in File 4 must point back to File 2, and the length of the LOB data stored in File 2 must match the actual length of the data stored in File 4.

Disk Error
If R:Scope is unable to read a portion of the hard disk, it reports the error as a physical read error.

5.3.4 Indexes

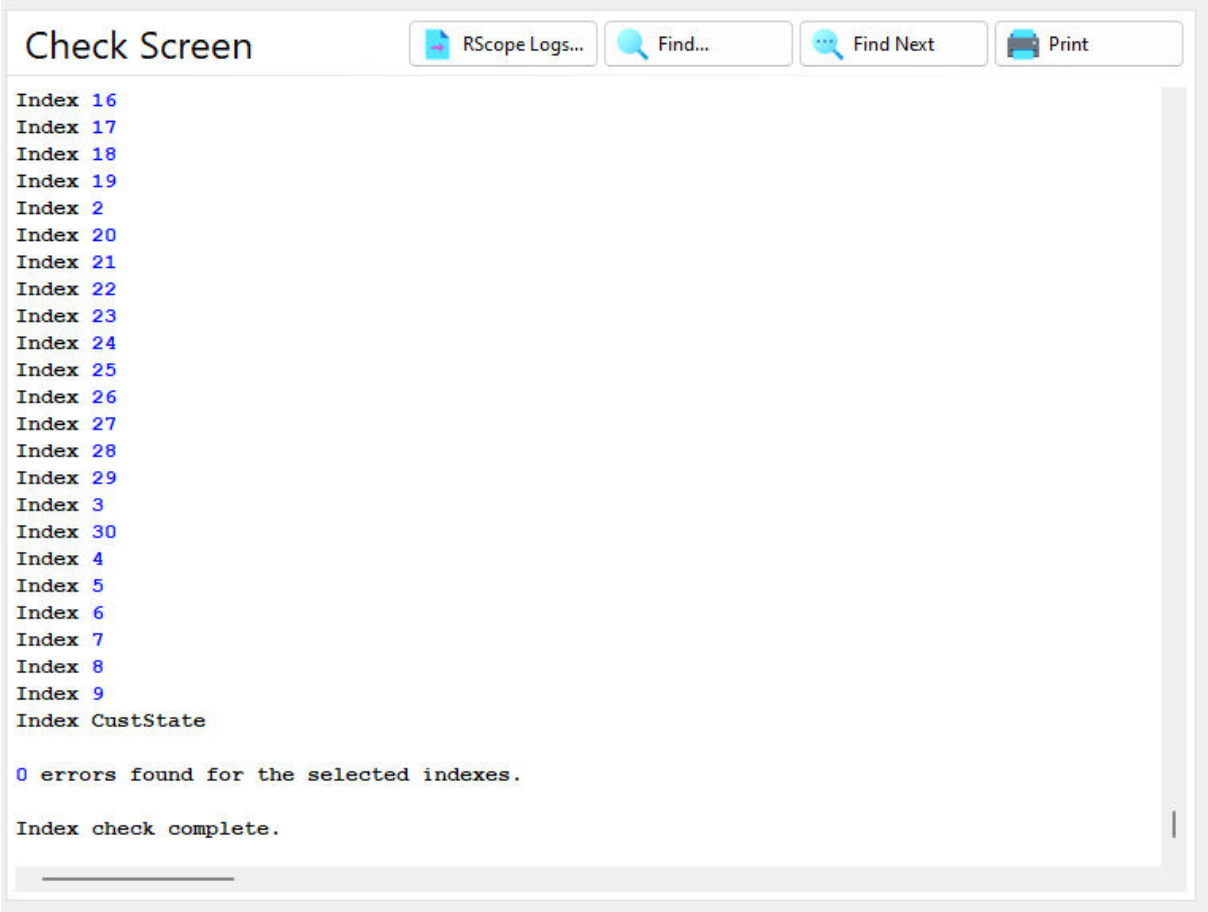
R:Scope can check database indexes, where a specific index or all indexes may be checked.

To check indexes:

1. Choose Check on the R:Scope menu and choose Indexes. R:Scope displays a list of indexes in the database.

2. With the mouse, select the index to check and select the "OK" button. To select more than one indexes, hold down the [Ctrl] key while selecting indexes. To check all the indexes, choose the "Select All" button. Then, select the "OK" button.

3. R:Scope displays the information on the Check Screen.



The process reviews each index (file 3) and walks through every pointer to the data file (file 2). An error is reported if:

- The pointer from the index does not point to a valid location in the 2 file
- The pointer from the index points to a valid location in the 2 file, but the row that lives there has been deleted
- The row count from walking the entire index does not match the row count for the table as stored in file 1

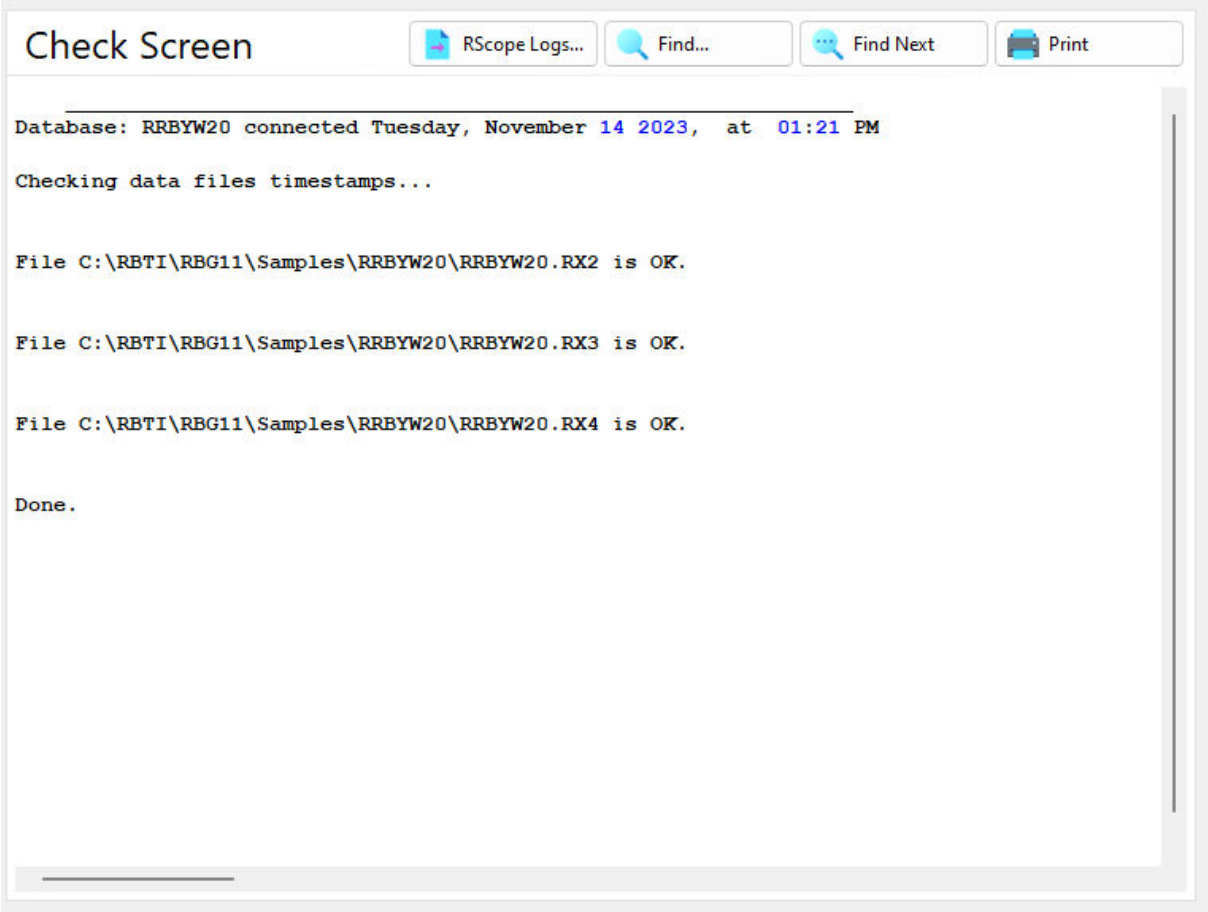
5.3.5 Database Files Timestamps

R:Scope can check the date and time stamps for the database files.

To check the file timestamps:

1. Choose Check on the R:Scope menu and choose Database Files Timestamps.

2. R:Scope displays the results on the Check Screen.



5.4 Fix

The options on the Fix menu allow you to edit the contents of the database structure and data files.

- Structure - lets you modify values in the Database Info, Settings, Table, and Column blocks in the structure file
- Data - gives you automatic and manual options for correcting pointer values in the data file
- Export - allows you to copy sections of the data file or the LOB file to another file

5.4.1 Structure

5.4.1.1 Database Info

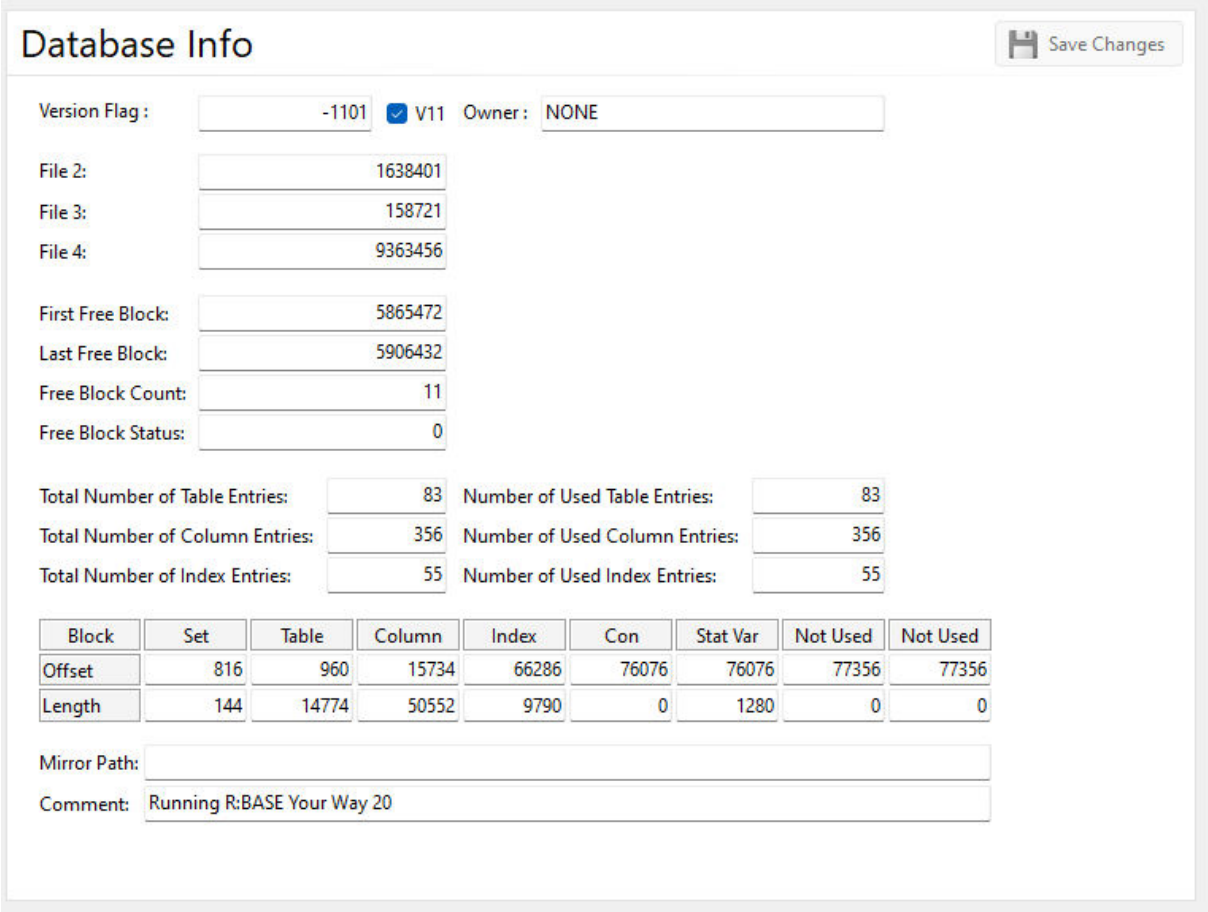
Although R:Scope allows you to edit the data in the Database Info block, the only items that you should attempt to correct using R:Scope are the version flag and the file size information.

The version flag is rarely incorrect. It is more common for the file size information to become incorrect. This can happen when a database is not properly closed before the computer is turned off. In that case, R:BASE might not have been able to update the file size information. If R:Scope finds an error in the length of the data file, correct it before checking the data file with R:Scope.

If R:Scope finds errors in the number of tables or the number of columns recorded in the Database Info section, try using R:BASE to correct the file. If the data appears correct when you view it using the SELECT or EDIT ALL commands, then R:BASE should be able to correct the file. Use the R:BASE BACKUP or UNLOAD commands to create a file that contains the database structure and data. Then use the RESTORE, RUN, or INPUT commands to recreate the database. In the process, R:BASE might be able to correct the structure file.

Caution: Before you use BACKUP or UNLOAD to correct the database structure, first use R:Scope to check the data file. Make sure that there are no structure-related errors (broken pointers) that would cause the loss of rows. Before using BACKUP or UNLOAD, make sure that R:BASE reserved words are not used as column or table names in the database.

To edit items seen in the Database Info section of the structure file:
1. Choose Fix from the R:Scope menu and choose Structure. R:Scope displays a submenu.
2. Choose Database Info. R:Scope displays a dialog box containing the information.
3. Use the mouse or press the [Tab] key or [Shift]+[Tab] keys to move to the item you want to change, and then edit the value.
4. Press [F2] or choose "Save Changes" when you have finished editing to save changes.



5.4.1.2 Database Settings

The structure file contains a block that stores settings for the database. You can change these settings using R:Scope if you want. Settings not included here can be changed only in R:BASE. R:Scope displays the settings as they are stored in the structure file. In some cases, this differs from the way the settings are displayed in R:BASE.

Characters
The settings are stored as they appear in R:BASE.

Date
The date formats are stored as they appear in R:BASE. The date sequence is stored using 1, 2, and 3 to show the position in the sequence of the day, month, and year for dates. 0 indicates no position in the sequence.

Time
The time formats are stored as they appear in R:BASE. The time sequence is stored using 1, 2, and 3 to show the position in the sequence of the hour, minute, and second for times. 0 indicates no position in the sequence.

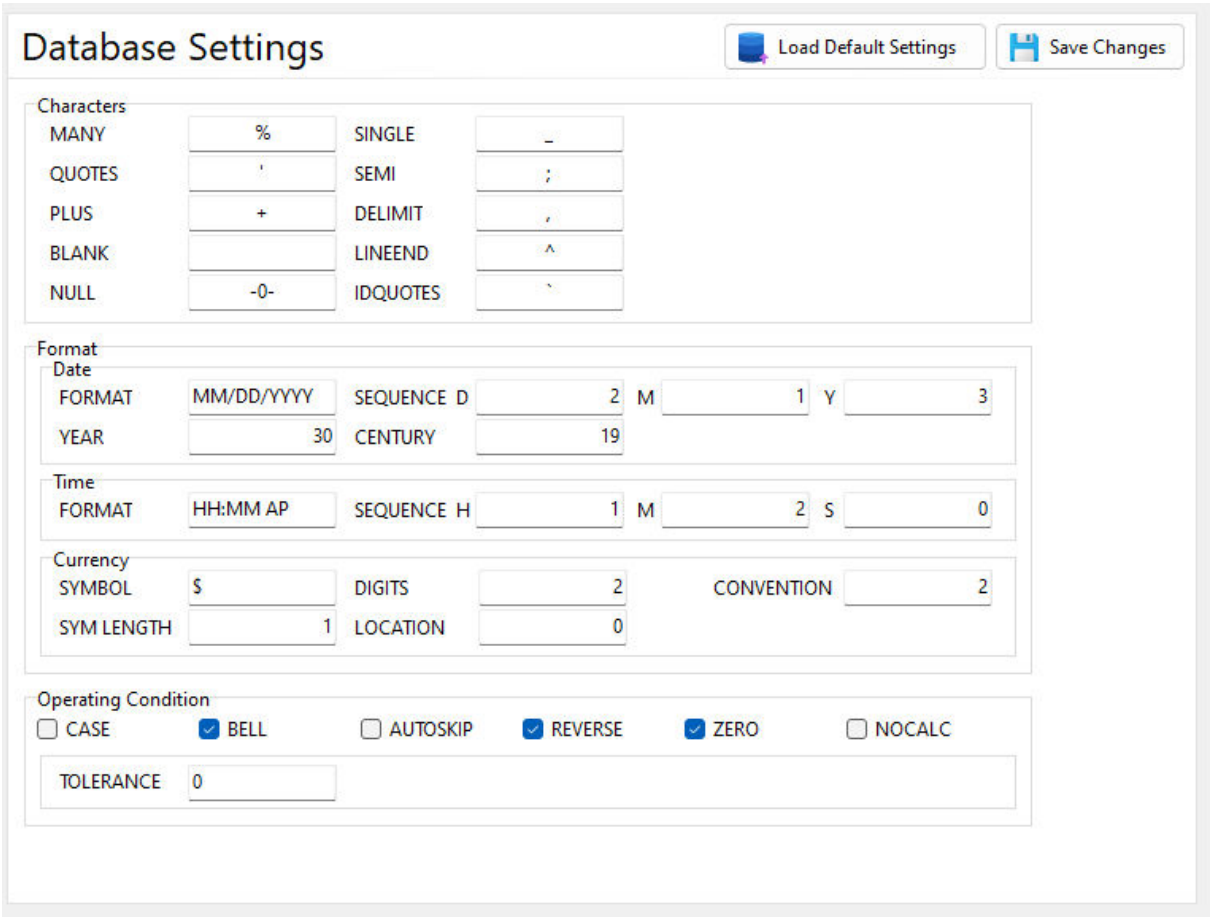
Currency
The currency symbol and number of digits in the currency sub-units are stored as they appear in R:BASE. The format, shown in R:Scope in the Convention field, is stored as 1,2, or 3. A 1 corresponds to the R:BASE A format, 2 to the B format, and 3 to the C format. Formats other than B require you to change the delimiter to a character other than a comma. The position of the currency symbol before or after the value is stored as 0 or 1. Use 0 to indicate a prefix; use 1 to indicate a suffix. The symbol length designates the number of characters in the currency symbol.

Operating Condition
The settings for CASE, BELL, AUTOSKIP, REVERSE, ZERO, and NOCALC are stored as they appear in R:BASE.

Tolerance
Tolerance is stored as it appears in R:BASE. The default value is 0.

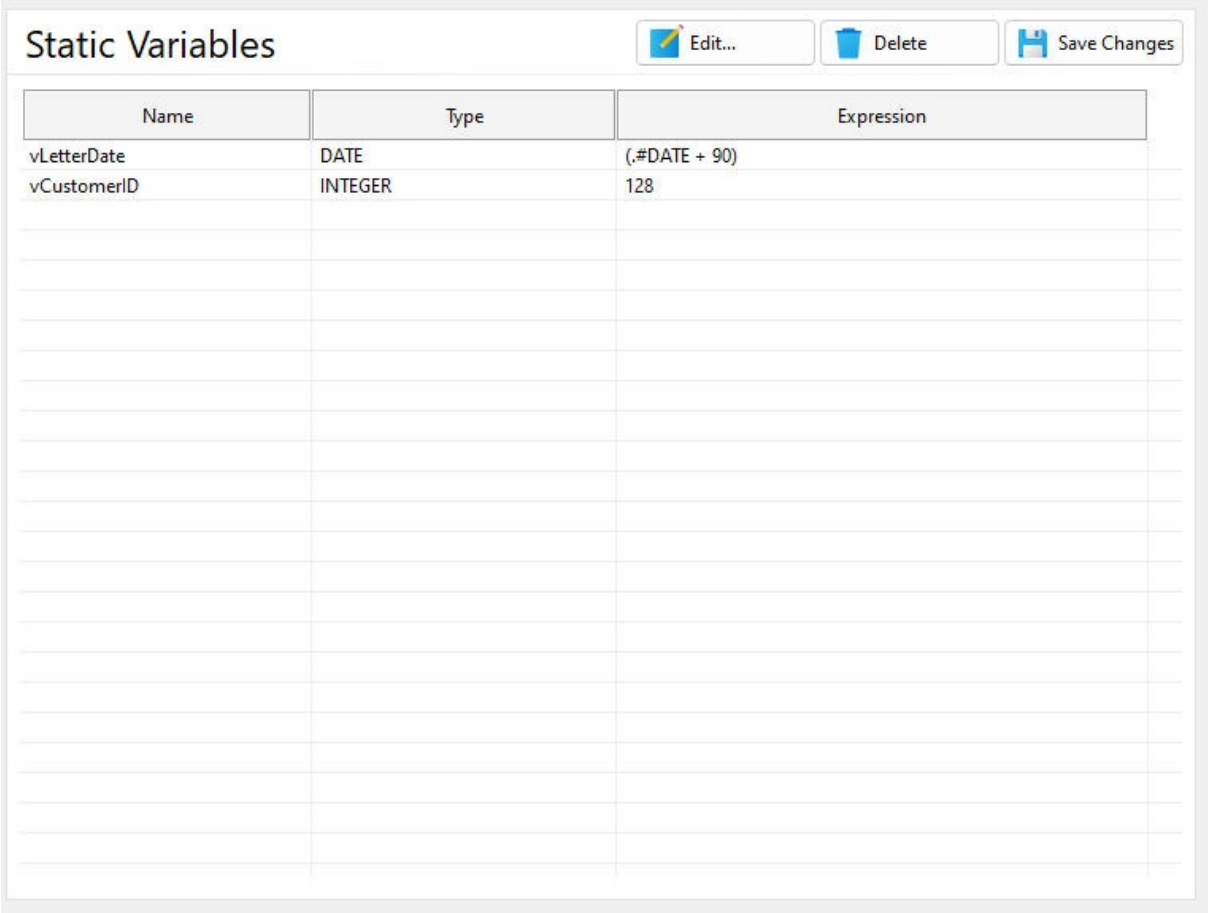
To edit the database environment settings:

1. Choose Fix from the R:Scope Menu and choose Structure. R:Scope displays a submenu.
2. Choose Database Settings.
3. Choose or press the [Tab] or [Shift]+[Tab] keys to move to the items you want edit.
4. Choose "Save Changes" or press the [F2] key to save changes when ready.



5.4.1.3 Static Variables

Checking the static variables, allows for the static variables to be altered, deleted, and saved. The Copy Expression(s) button allows for the selected static variables to be placed in the clipboard.



5.4.1.4 Tables

If checking the database structure file shows errors in the Table block, you can correct the block with the Fix option.

Although R:Scope allows you to edit all the data in the Table block of the structure file, in practice the only items you should change are the values for the number of rows in tables and the values for the starting and ending pointers for rows in tables. For example, when you are fixing erroneous pointers in the data file, you might need to discard the rows whose pointers are incorrect. Once you have done so, the table will contain fewer rows and you should correct the value for the table's number of rows to reflect this change in the database.

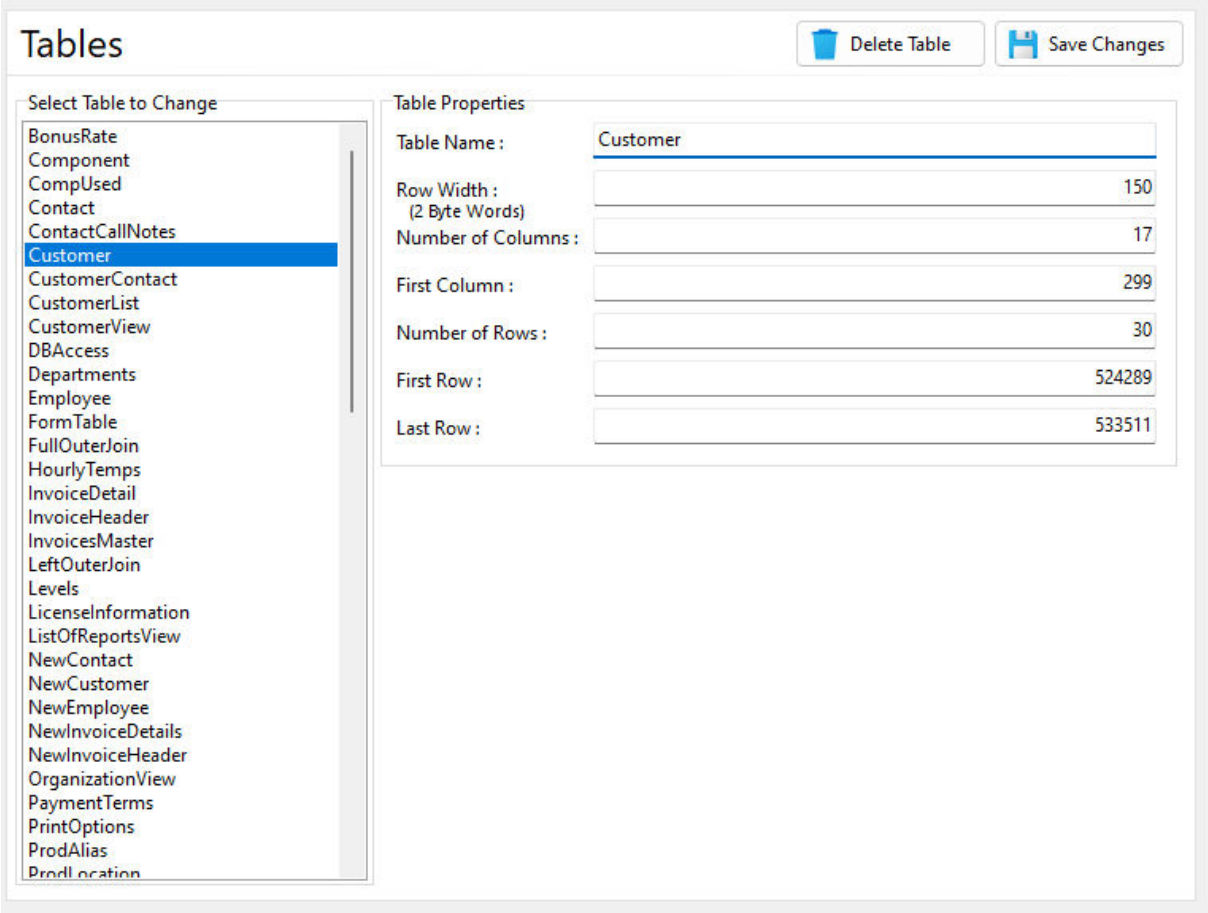
If R:Scope finds errors in the row width, the number of columns in the table, or the order of the table's first column in the database, try using R:BASE to correct the file. If the data appears correct when you view it using the SELECT or EDIT ALL commands, then R:BASE should be able to correct the file. Use the R:BASE BACKUP or UNLOAD commands to create a file that contains the database structure and data. Then use the RESTORE or INPUT commands to recreate the database. In the process, R:BASE might be able to correct the structure file. For more information about these commands, refer to your R:BASE manuals.

Caution: Before you use BACKUP or UNLOAD to correct the database structure, use R:Scope to check the data file. Make sure that there are no errors that would cause the loss of rows. Also before using

these commands, make sure that R:BASE reserved words are not used as column or table names in the database.

To fix items in the Table block of the structure file:

1. Choose Fix from the R:Scope Menu and choose Structure. R:Scope displays a submenu.
2. Choose Tables. R:Scope displays a list of tables in the database.
3. Select the name of the table whose information you want to edit. R:Scope displays a dialog box containing the information.
4. Choose or press the [Tab] or [Shift]+[Tab] keys to move to the items you want to edit.
5. Select the "Save Changes" button or press [F2] when you have finished editing to save changes.



5.4.1.5 Columns

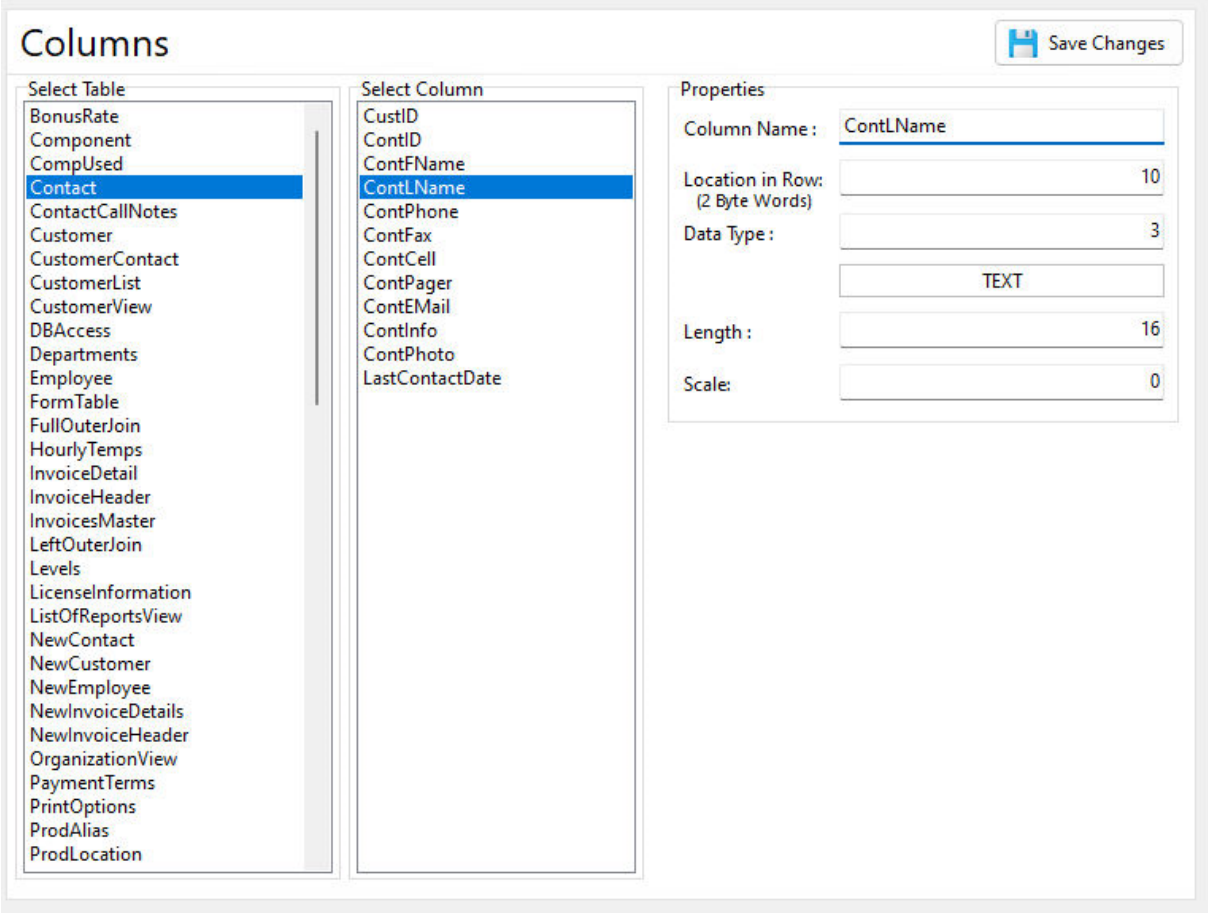
Although R:Scope allows you to edit all the data in the Column block of the structure file, you should attempt to repair it in R:BASE and use R:Scope only as a last resort.

The column name, data type, and Index block information can all be changed in R:BASE. Because changing the location of the column in the row or the length of a column affects columns appearing later in the table, errors should be repaired in R:BASE. If the data appears correct when you view it using the SELECT or EDIT ALL commands, then R:BASE should be able to correct the file. Use the R:BASE BACKUP or UNLOAD commands to create a file that contains the database structure and data. Then use the RESTORE, RUN, or INPUT commands to recreate the database. In the process, R:BASE might be able to correct the structure file. For more information about these commands, refer to your R:BASE Help.

To fix items in the Column block of the structure file:

1. Choose Fix from the R:Scope Menu and choose Structure. R:Scope displays a submenu.

2. Choose Columns. R:Scope displays a list of tables in the database.
3. Select the name of the table containing the column whose information you want to edit. R:Scope displays a dialog box, shown below, containing information about the columns in the table.
4. Select the name of the column you want to edit. R:Scope displays a dialog box, shown below, containing information about the column.
5. Select the field or press the [Tab] or [Shift]+[Tab] keys to move to the items you want to edit.
6. Select the "Save Changes" button or press [F2] when you have finished editing to save changes.



5.4.1.6 Fixing Index Structure

If you are having trouble retrieving or adding data and suspect the problem is a bad index, check the database data and structure with the AUTOCHK command in R:BASE, or with R:Scope.

If you have errors in the data or structure, correct the errors and execute a PACK ALL command in R:BASE. After correcting errors with R:Scope, you must execute PACK ALL to remove the unused space from File 1, rebuild the File 2 data pointers, and rebuild all the indexes. After the PACK ALL, retry the operation that was not working. If there are no errors in the data or structure, retrieve the missing rows in R:BASE with an indexed data retrieval and a non-indexed data retrieval. For example:

```
SELECT transid FROM transmaster WHERE transid = 5000
```

uses the index on the *transid* column to retrieve the data.

```
SELECT transdate FROM transmaster WHERE transid = (5000)
```

sequentially searches all the rows in the table and does not use indexes.

Select a non-indexed column and put parentheses around the comparison value to force R:BASE to do a non-indexed data search. Rebuild the indexes if the non-indexed retrieval finds data but the indexed retrieval does not. Create a single index by using the DROP INDEX and then CREATE INDEX commands. Use the PACK KEYS command to rebuild all indexes. Retry the operation that was not working. For more information about these commands, refer to the R:BASE Help.

5.4.2 Data

5.4.2.1 Automatically Fixing Data File Pointers

The automatic fix option rebuilds pointers for a table. When R:Scope automatically fixes a table, it moves through the data file a row at a time using the data file's pointers.

When R:Scope finds one or more rows whose pointers appear incorrect, R:Scope searches for the next row whose pointers appear to be correct and whose row width matches the width recorded for the table in the Table block of the structure file. When R:Scope finds a row that appears valid, it bypasses those rows whose pointer values are incorrect and links the valid row to the table by editing the pointer values. As a last step, R:Scope edits the structure file to correct the number of rows recorded for the table and, if necessary, the ending pointer.

R:Scope tells you how many rows it will discard when repairing the table. If automatically fixing the table will cause the loss of many rows of data, you might want to stop the operation and review rows that will be discarded. It is possible that the data is still sound, though the pointers to it are damaged.

To find the damaged rows in the data file, use the error address information R:Scope generated when checking the data file with the Check option. Then use R:Scope's Display option to examine rows in the data file. If data in the rows is undamaged, you might want to fix the table manually.

For more information about displaying data from the data file, see Seeing Data File Contents.

There are limits to the use of the automatic fix option. Do not use the automatic fix option on a table when:

- The table contains a column with a NOTE data type.
- The table has the same row width as another table in the database. This could cause R:Scope to cross-link the tables. To see the row widths for the tables in the database, choose Display on the R:Scope Menu and choose Table Structure.

To automatically fix a table:

1. Choose Fix on the R:Scope menu and choose Data. R:Scope displays a submenu.

2. Choose Automatic. R:Scope displays a list of tables in the database.

3. Select the name of the table you want to fix. R:Scope can only repair one table at a time. R:Scope determines how many rows have damaged pointers and displays a dialog box giving you the option to continue or stop the operation. Choose Yes to continue, No to stop.

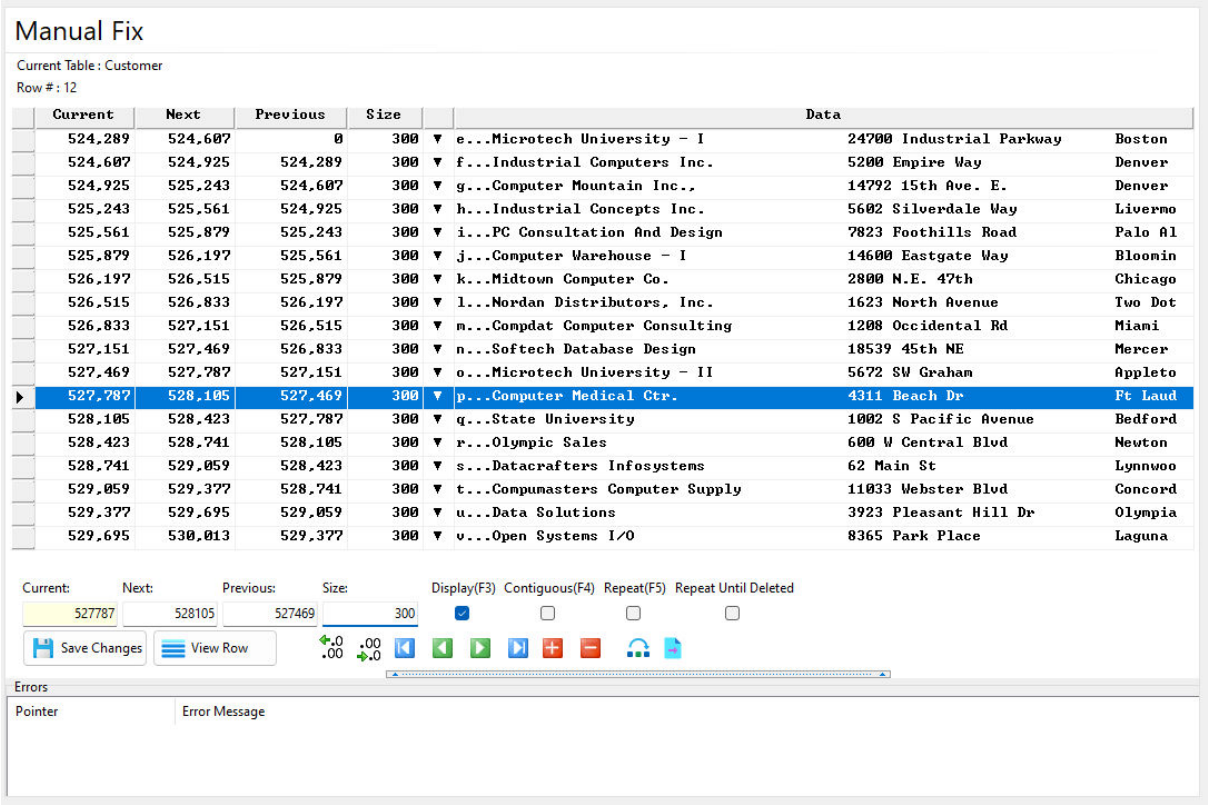
4. If you choose "Yes" to continue the repair of the table, R:Scope repairs the table and reports the changes to the table's pointers on the Check Screen.

5.4.2.2 Manually Fixing the Data File

The Manual option allows you more control over the data file than the Automatic option. By using the Manual option, you can:

- Rebuild pointers in the data file
- Find and undelete rows of deleted data

Note: When you are working with the data file on the Fix: Data screen, display the Check Screen by selecting the "Check" tab, or press [Page Down] and return with [Esc]



To manually fix a table:

1. Choose Fix on the R:Scope menu and choose Data.
2. Choose Manual.
3. Select the name of the table you want to fix.

R:Scope displays the first row of information for the chosen table or database on the Fix: Data screen. Press the [Down Arrow] key to display additional rows. The screen grid will highlight the current row. For each row, R:Scope displays the following information:

Current
R:Scope displays the current row's address in bytes. This is the physical location for the start of the row in the data file. It is the count in bytes from the beginning of the file. Because the value measures the distance from the start of the file that the row is displaced, it is sometimes called an *offset.* The current address is a count of bytes, not a value stored in the data file like a pointer.

Next
R:Scope displays the row's next row pointer. This 8-byte value shows the physical location in bytes of the table's next row in the file. The next row pointer shows the current address for the table's next row. The next row pointer for the last row in the table is always set to 0.

Previous
R:Scope displays the row's previous row pointer. This 8-byte value shows the physical location in bytes of the table's previous row in the file. The previous row pointer shows the current address for the table's previous rows. The previous row pointer for the first row in the table is always set to 0.

Size
R:Scope displays the size of the row. This 2-byte value shows the size of the row in bytes. The row size is the sum of the column widths that make up the row. For all tables except those that contain NOTE columns, the size of each row will be the same. The row size reported by R:Scope accounts for the row's

data only. The actual physical size of the row is 18 bytes larger because it includes the two 8-byte next and previous row pointers and the 2-byte row size.

Contiguity
The untitled column to the right of the size column indicates whether a row is contiguous with the next row. A contiguous row is located directly before the next row in the data file. A square indicates a contiguous row, and a triangle indicates a noncontiguous row. For contiguous rows, the next row pointer should always be the value of the current address plus the size of the row plus 18 bytes. The 18 bytes are the row's two 8-byte next and previous pointers plus the row size stored in 2 bytes.

R:Scope can warn you when you reach a row that is not contiguous. You can toggle this feature on and off by highlighting Set on the R:Scope menu and choosing Contiguous or by pressing [F4]. When contiguity checking is on, R:Scope stops at noncontiguous rows and displays an error message.

Data
R:Scope displays the first 30 characters of the row's data as it is stored in the data file. Only TEXT and NOTE data will be comprehensible.

5.4.2.2.1 Moving To and Displaying Rows

R:Scope displays a row from the data file when you move to the row, and indicates the current row by displaying a block shaped marker next to the row in the first column.

When you are moving from row to row in the data file, R:Scope displays a message when an error is encountered. For example, if the next pointer value is out of range, R:Scope will let you know.

R:Scope provides these ways to move to the rows in the file:

Next and Previous Row Pointer
The [Down Arrow] and [Up Arrow] keys move from row to row within a table by using the next and previous row pointers that are stored with the current row. You can also use the green buttons on the bottom toolbar. Hovering your cursor over the arrows and buttons will display a hint of the button. The next or previous row in a table might not be physically contiguous within the data file.

R:BASE marks the beginning and end of tables by assigning a value of 0 to the previous row pointer at the start of a table and a value of 0 to the next row pointer at the end of a table. R:Scope displays a message to tell you that you have reached the last row in a table when it reaches a 0 next pointer value.

You can also use the scrolling button on a mouse to move from row to row using the row pointers. To move to the next row, scroll down on the scroll button. To move to the previous row, scroll up.

Structure File Starting and Ending Pointers
The [Home] and [End] keys move to the first and last row in a table by using the starting row and ending row pointers that are stored in the structure file.

Specified Address
The [F7] key displays a dialog box to enter an address value. When you enter a value, R:Scope moves to that address and displays the data there as a row. If the address you specified does not mark the start of a row, the data R:Scope displays, including that for pointers, will appear incorrect or garbled. Use this option, for example, when you want to return to a row that you have passed and the current pointer is out of range. Enter the current address for the row to return to it. You can also use the mouse to return to a passed row that is out of range as long as it is still visible on the screen. Simply click on the row.

5.4.2.2.2 Keys Available from the Fix: Data Screen

The table below lists the hot keys available when you are manually fixing data.

Key	Action
[Down Arrow]	Moves to next row in the current table.
[Up Arrow]	Moves to previous row in the current table.
[Home]	Moves to first row in the current table.
[End]	Moves to last row in the current table.
[Delete]	Deletes the current row.

[Insert]	Undeletes the current row.
[Ctrl] + [Right Arrow]	Shifts the data displayed as a row one byte to the right.
[Ctrl] + [Left Arrow]	Shifts the data displayed as a row one byte to the left.
[Page Down]	Displays the Check Screen. Press [Esc] to return to the Fix: Data screen.
[F3]	Turns display of the rows scrolling off and on. Used to move through rows quickly without displaying them to the screen. The default is display ON.
[F4]	Turns contiguity warning on and off. When contiguity warning is ON, R:Scope warns you when rows are not contiguous in the data file. The default is OFF.
[F5]	Repeats the next command. For example, while editing data, press [F5] and then [Down Arrow] to continue displaying rows. Press [F5] again to stop.
[F6]	Adjusts shifted row offsets. R:Scope displays a dialog box to enter the address range to adjust and the offset amount by which to change each row.
[F7]	Moves to the specified address. R:Scope displays a dialog box in which you can enter a value. The default value is the current row's address. R:Scope moves to that address and displays the data there.
[F8]	Sets the row counter. For example, if you want to count the number of rows in a section of the data file, set the row counter to 1 and R:Scope will count the rows as you move through them.
[F9]	Views the current row in R:BASE format.
[F10]	Finds the next row of data using the search method set on the Set menu. The default search method is Smart.

5.4.2.2.3 Using Repeat

R:Scope's Repeat option can help you easily display all of the rows in a table.

1. After you display the first row in the table, switch repeat on by selecting the Repeat check box in the Manual Fix screen

2. Press [Down Arrow]. R:Scope displays rows until reaching the end of the table or until you uncheck the Repeat check box to switch repeat off. You can also use this method to:

- Move up through the previous rows in a table by pressing [Up Arrow].
- Undelete or delete many rows by pressing and holding down [Insert] or [Delete].

5.4.2.2.4 Using Repeat to Deleted

R:Scope's Repeat to Deleted option can help you easily move to deleted rows in a table.

1. After you display the first row in the table, switch repeat to deleted on by selecting the Repeat to Deleted check box in the Manual Fix screen

2. Press [Down Arrow]. R:Scope displays rows until reaching the first deleted row encountered or until you uncheck the Repeat to Deleted check box to switch the setting off. You can also use this method to:

- Move up through the previous rows to a deleted row in a table by pressing [Up Arrow].
- Move down again to further deleted rows in the table.

When a deleted row is encountered, the "View Row" button will allow you to review the data for what was deleted, and perhaps restore the row.

5.4.2.2.5 Toggling the Display Off and On

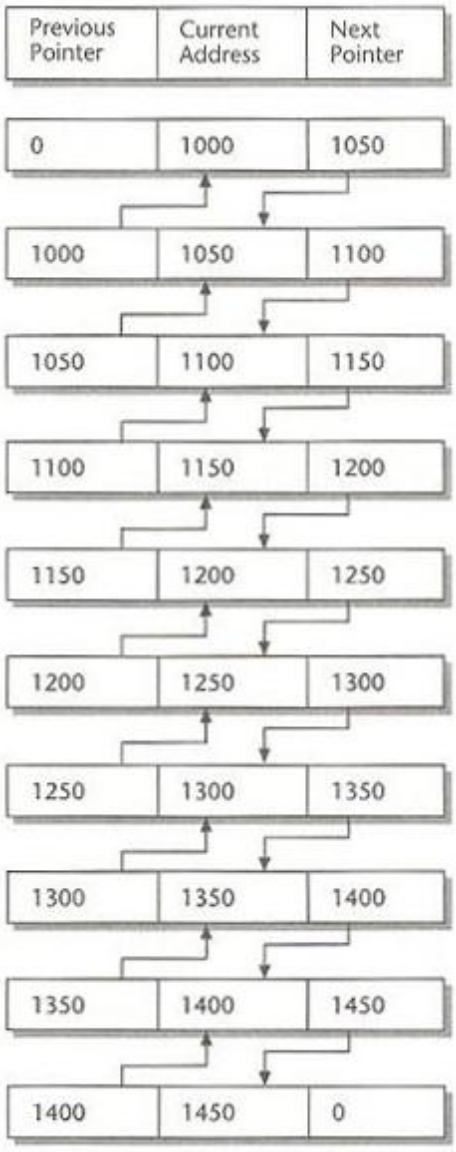
When you are moving through many rows of a table using the row pointers and the repeat function, R:Scope can move more quickly through the table if rows are not displayed on the screen. Because R:Scope will stop when an error is encountered and then display the row when you press any key to continue, this feature is useful when you are looking for errors in large tables.

To toggle the display of rows off, select the Display check box or press the [F3] key. Then set repeat on and move through the rows. R:Scope will move through the rows until it finds an error, finds a noncontiguous row when contiguity checking is on, or reaches the end of the table.

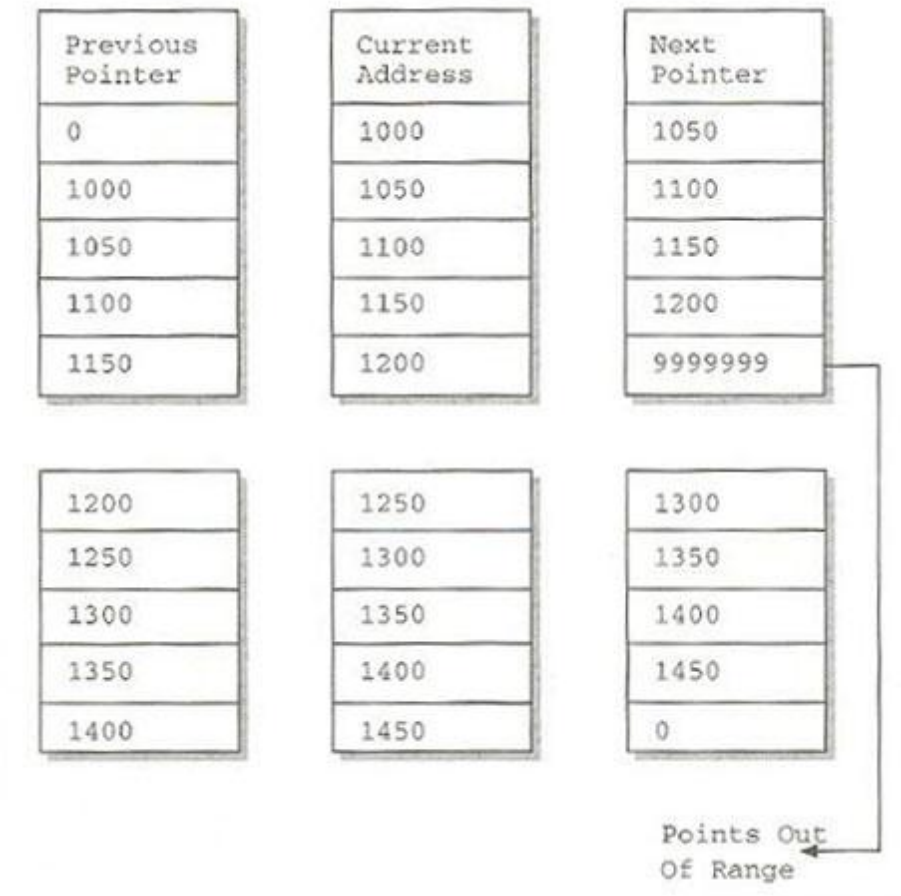
5.4.2.2.6 Editing Pointers

The most common damage to a database occurs when the pointers and data for a row or group of rows in the data file become damaged. Normally, the rows in the file are linked by their pointers. For each row in the table, the next row pointer indicates the data file location of the row after the current address and the previous row pointer indicates the data file location of the row before the current address. Each row is chained to the row before it and the row after it by its pointers.

The illustration shows this relationship for a simple table of 10 rows whose first row is 1000 bytes into the data file. The physical size of each row is 50 bytes: 22 bytes of data plus 8 bytes for the next and previous pointer values plus 2 bytes for the row size information. For clarity, previous pointer values are shown to the left of the current address values here. Note that previous pointer values appear to the right of the next pointer values on the R:Scope Fix: Data screen.



When one or more pointers become damaged in the data file, a break in the chain occurs. R:BASE will not be able to find those rows in the table after the break. In the simple case illustrated here, the next row pointer in the row at address 1200 is damaged and indicates that the next row is located 9,999,999 bytes into the data file.



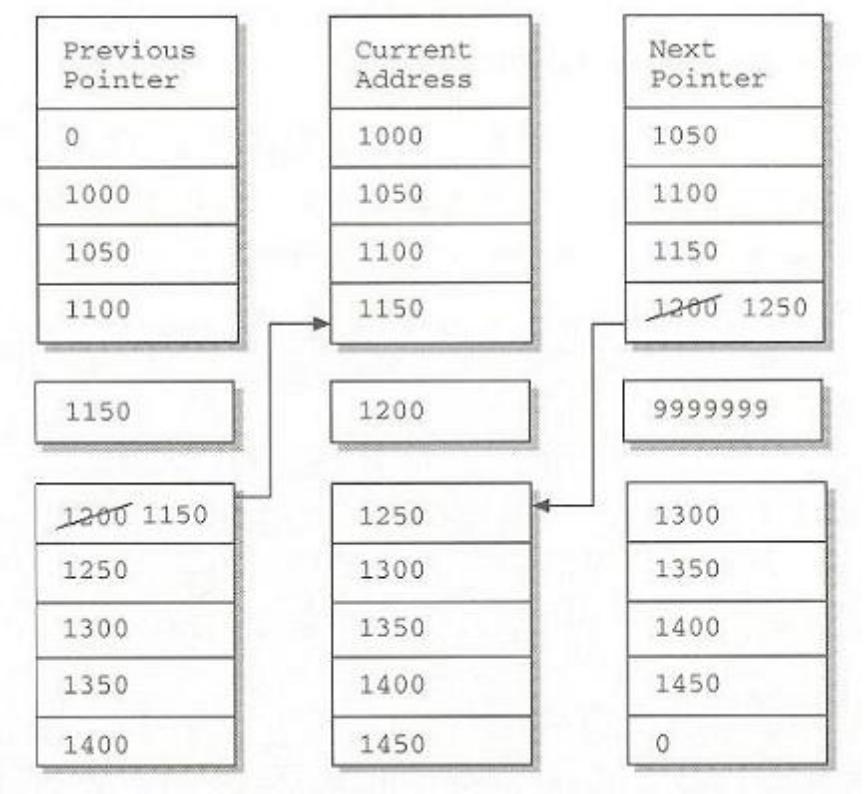
To repair the data file, you need to bypass the damaged group of rows by linking the correct row before the damaged block to the correct row that follows the damaged block.

To link the rows:

- Change the next pointer value for the row before the damaged block to the value of the current address for the row after the damaged block.
- Change the previous pointer value for the row after the damaged block to the value of the current address for the row before the damaged block.

To repair the data file in the example, you would bypass the row at address 1200 by changing the next row pointer at address 1150 to 1250 and the previous row pointer for the row at address 1250 to 1150, as shown in the next image.

Although R:Scope allows you to edit the row size as well as the pointer values for a row, in general you should not need to change the row size.



To edit the pointers in a row:

1. Select the row whose pointers you want to edit.
2. The pointer values for that row will appear in the fields for the next and previous row pointers as well as the row size at the bottom of the screen

- To move between the fields, press the [Tab] or [Alt]+[Tab] keys.
- To move within a field, press the [Right Arrow] or [Left Arrow] keys.
- If, before you move off a field, you want to cancel your changes and restore the original value, select another pointer row and the original value will return

3. Select the "Save Changes" button or press [F2] to save your changes.

Note: Before editing the pointers, it is recommended that you make a backup copy of the database files using a operating system COPY method.

5.4.2.2.7 Restoring and Deleting Rows

You can restore deleted rows to a table or delete existing rows from a table by using R:Scope. If you restore or delete rows, you will need to rebuild the indexes for the table.

Restoring Rows

R:BASE marks a row as deleted by making its row size a negative number.

To restore a deleted row:

1. Move the block cursor to the row you want to undelete.
2. Select the red button with the plus (+) or press the [Insert] key to undelete the row. R:Scope restores the row and moves to the next wow in the table. The row size will now be a positive number.

If you have many rows to restore in a table, you can use R:Scope's repeat feature after holding down the [Insert] key to move through the table restoring rows as it goes.

When you undelete a row that has LOB data, the LOB data is not restored. The LOB column is set to NULL as soon as the row is deleted, and the blocks from File 4 are added to the free block list. You might be able to find and export the LOB data by examining the LOB file and then exporting data from the free block list.

Deleting Rows To delete an existing row:

1. Move the block cursor to the row you want to delete.
2. Select the red button with the minus (-) or press the [Delete] key to delete the row. R:Scope deletes the row and moves to the next row in the table. The row size will now be a negative number.

If you have many rows to delete in a table, you can use R:Scope's repeat feature after holding down the [Delete] key to move through the table deleting rows as it goes.

5.4.2.2.8 Restore After All Rows Deleted

If all rows are deleted from a table, the entire number of rows may be restored. If partial records are deleted, see Restoring and Deleting Rows.

NOTE: This type of recover cannot restore BLOB data. If there are VARCHAR or VARBIT data type columns in the table, the column data will not be restored.

To restore all rows, after all rows are deleted for a table:

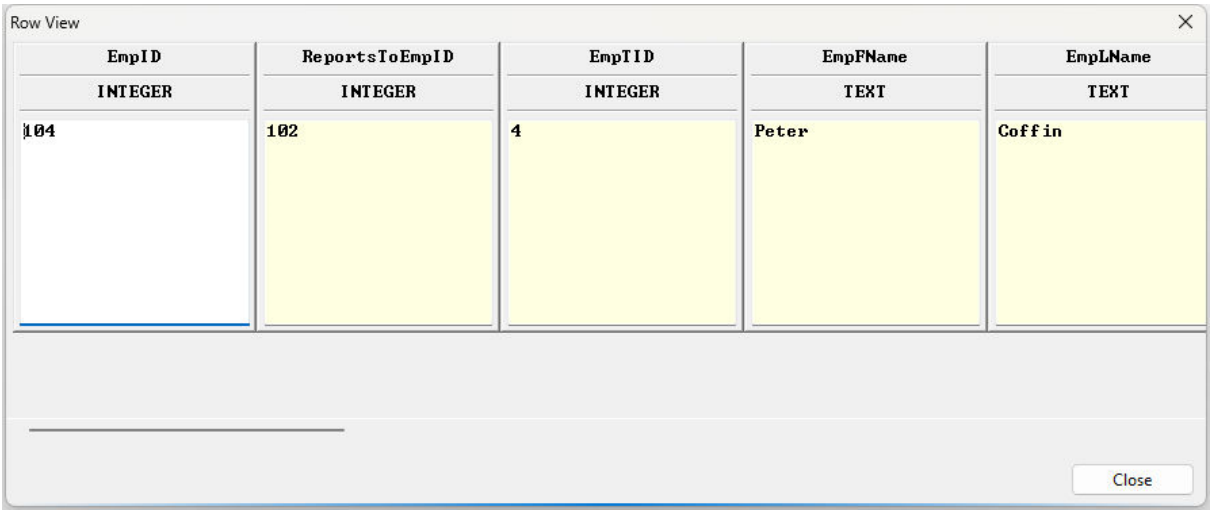
1. Select "Display" > "Table Structure" from the menus, and locate the table where all rows are to be restored
2. Record the "Starting Pointer" value on paper, or elsewhere
3. Select "Fix" > "Data" > "Manual..." from the menus
4. Select any one of the displayed tables and choose OK
5. Select the "Jump to...[F7]" button, to jump to a pointer address
6. In the displayed dialog, enter the "Starting Pointer value recorded in step 2
7. With "Display [F3]" unchecked/off and "Repeat [F5]" checked/on, select the "Next" button, for R:Scope to travel to the end of the table pointer. Do not use the "Last" button as R:Scope will go to the last pointer for the opened table.
8. Once at the end, R:Scope will display a message stating the end of the table has been reached. The table name in the dialog can be ignored.
9. Record both the "Row #" value (listed under "Current Table")
10. Next, record the ending pointer value (listed within the "Current" field)
11. Select "Fix" > "Structure" > "Tables" from the menus, and select the table where all rows are to be restored
12. Within the "Number of Rows" field enter the "Row #" value recorded in step 9. (listed within the "Current" field.
13. Within the "Last Row" field enter the ending pointer value recorded in step 10
14. Select the Save Changes button

R:Scope may now be closed, and R:BASE may be connected to the database to verify the restore of the deleted rows.

5.4.2.2.9 Seeing the Data In a Row

To see the data stored in a row organized by columns in R:BASE format, move to the row and select the "View Row" button or press the [F9] key. You can also click the data column for the row with the mouse. R:Scope displays data for the row column by column in a dialog box. R:Scope, including TEXT and NOTE columns, displays a maximum of 15 characters of data for each column. If there are more columns in the row than the screen can display at once, use the [Right Arrow] and [Left Arrow] keys to scroll columns into view. You can also click the scroll bar at the bottom edge of the box to move columns into view.

For LOB data (VARBIT and VARCHAR columns), R:Scope displays the file type (FRM, BMP, and so forth), the starting addresses of the LOB data in File 4, and the length in bytes of the data in File 4.



5.4.2.2.10 Searching for Rows

When you are repairing a table whose pointers are broken, you might need to find rows whose locations are dispersed throughout the data file. R:Scope can help you find rows in the data file by searching for them. R:Scope offers three search methods: smart search, search by pointer, and search by data.

To search for a row:

1. Choose "Set" from the tool bar, and choose Search. R:Scope displays a submenu of search types.

2. Select the type of search. The default search type is Smart Search. Once you set a search type, it remains set for that session until you change it. If you select Pointer or Data, R:Scope prompts you to enter search criteria.

3. Press [F10] when you are on the Fix: Data screen to execute the search. R:Scope starts searching from the current row in the data file to find the next row in the file that fits the search criteria.

Smart Search

Smart Search automatically determines parameters for the next row's pointers and width. When you start a Smart Search, R:Scope moves through the data file byte by byte looking for numbers that fall in the correct range. When R:Scope finds data whose pointers and width are in range, it displays it as a row. See the table below for a list of Smart Search parameters.

Smart Search Parameters

Parameter	Value
Minimum next row pointer	Equals offset + 18 bytes + minimum row size. R:Scope recalculates this value each time it moves to the next byte in the file as the offset changes.
Maximum next row pointer	End of table pointer taken from the structure file.
Minimum previous row pointer	Start of table pointer taken from the structure file.
Maximum previous row pointer	Equals offset - 18 bytes - the minimum row size. R:Scope recalculates this value each time it moves to the net byte in the file.
Maximum row size	Row size for the table unless the table contains a NOTE column. The maximum size of a row that contains one or more NOTE columns is 32,768 bytes.

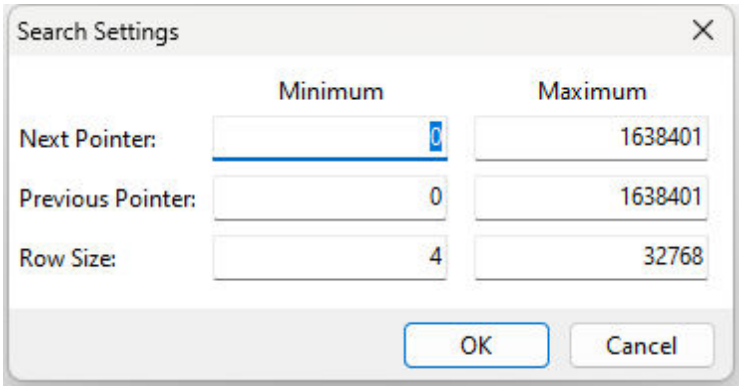
Minimum row size	Row size for the table unless the table contains a NOTE column. The minimum size of a row that contains one or more NOTE columns equals the sum of the widths for each column. NOTE columns are counted as 4 bytes.

Pointer Search

Pointer search allows you to set the parameters for the next row's pointers and width. When you start the search, R:Scope moves through the data file byte by byte looking for numbers that fall in the ranges you specified. When R:Scope finds a row whose pointers and width are in range, it displays the row.

When you set R:Scope to perform a pointer search, R:Scope displays a dialog box in which to enter the parameters for the search. See the table below for a list of initially set pointer search parameters.

One strategy of searching for rows using a pointer search is to set the maximum and minimum row size to match the row size in the table you are working on. Provided that no other table in the database has rows of the same width, this will narrow the search to only those rows belonging to the table. This method is not as suitable for tables containing NOTE columns because such tables have rows of varying sizes.



Pointer Search Parameters

Parameter	Value
Minimum next row pointer	0
Maximum next row pointer	Data file size taken from the structure file
Minimum previous row pointer	0
Maximum previous row pointer	Data file size taken from the structure file
Maximum row size	32,000 bytes
Minimum row size	4 bytes

Data Search

Another way to locate a row for a table in the data file is to search for a value that you know exists, Use the data search option to do this. When you start the search, R:Scope moves through the data file byte by byte looking for the value you specified. When R:Scope finds the value, it displays a dialog box showing the pointer value for that row. Press [Enter] to display the row at that pointer.

When you set R:Scope to perform a data search, R:Scope displays a submenu listing data types. Select the data type of the value based on the data type of the value's column. R:Scope then prompts you to enter the value. If you are searching for a DATE value, the format must be MM/DD/YYYY. If you are searching for a TEXT value, make sure that you enter the case of letters accurately. R:Scope distinguishes between upper and lower case when searching for TEXT values.

When possible, use TEXT values for searches. When searching, R:Scope matches the search value with the first equivalent value found in the data file. When the search value is numeric, it is more likely that

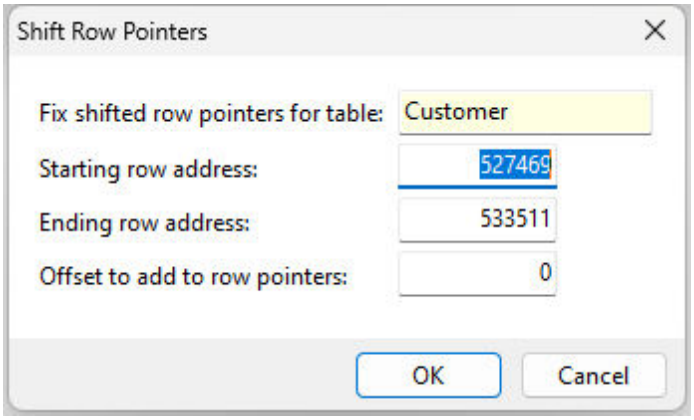
R:Scope will find a match that is actually part of a pointer or a series of numbers belonging to two columns. Using TEXT values reduces this possibility and the search is more likely to find a valid row.

5.4.2.2.11 Adjusting Row Pointers

If there is an error in the computer's file allocation table or if an undelete utility placed portions of File 2 in the wrong location, row pointers could be offset within File 2. The next and previous pointers for a block of data could be correct, but the location of the data within File 2 (the current address) is wrong.

To see if a block of data has been shifted, press [Tab]. If you can correctly find the next row by pressing [Tab] but you can't find the next row based on the next pointer when you press [Down Arrow], you probably have a group of shifted rows.

To correct this shift, press [F6] from the Fix: Data: Manual screen for this table. The default value for the starting row address is the current row address when you pressed [F6]. If this is not the address of the first shifted row, change the value.



The default ending row address is the address for the last row in the table. If the last shifted row is not the last row in the table, change this value to the address of the last shifted row.

The offset is 0 by default. Calculate the offset using this formula:

Offset = (next row address) - (current row address) - (row size) - 18

The offset value can be positive or negative. If the result of the calculation is positive, change the offset to the negative of the result. If the result is negative, change the offset to the positive of the result.

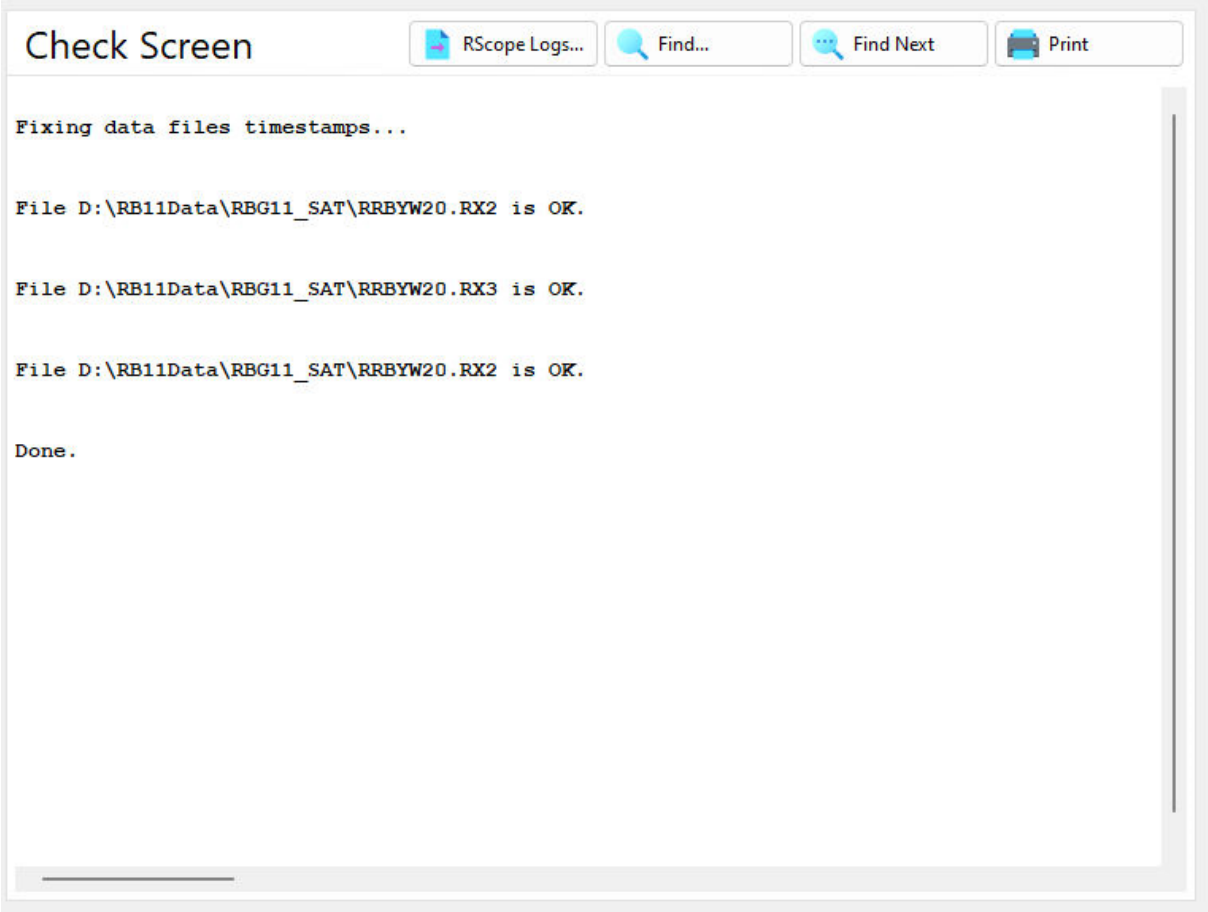
Press [F2] to execute the adjustment. If R:Scope senses that the offset value is incorrect, an error message is displayed. Correct the offset and press [F2] to retry the adjustment. R:Scope reports the number of rows modified.

5.4.3 Database Files Timestamps

R:Scope can fix the date and time stamps for the database files.

To fix the file timestamps:

1. Choose Fix on the R:Scope menu and choose Database Files Timestamps.
2. R:Scope displays the repair results on the Check Screen.

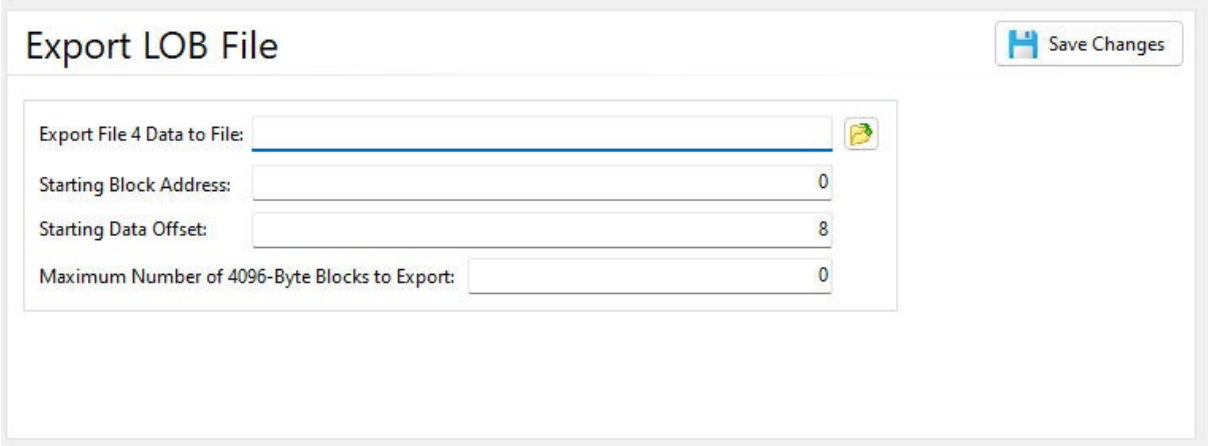


5.4.4 Exporting LOB Data

Export is for LOB data that has been orphaned, and is used to copy data from the LOB file to a disk file where you can recover or re-enter the data using R:BASE. You specify the starting address for the LOB data and the number of blocks to copy. You might need to copy the LOB data out of the LOB file when you have damaged File 2 pointers or deleted rows.

Press [F9] in the Fix: Data: Manual screen to see the file type, address, and length of the LOB data for a row. The address shown can be used as the starting address for the export. The length tells you how many blocks to export. You can also find an LOB address by displaying the LOB file or charting the LOB file.

After choosing Fix: Export, R:Scope displays a dialog box where you enter the filename, export mode, address, offset, and number of blocks.



The specified file is created by R:Scope. The file extension should match the file type to be exported, such as .BMP. The export mode is either TEXT for VARCHAR data, or BINARY for VARBIT data. The starting address is the beginning of the LOB data chain to be exported. You can specify to skip the first 8 bytes of LOB data, which is the header information that points back to File 2. The number of blocks is the maximum number of blocks to be exported. R:Scope will read and export blocks until the end of the LOB data chain is reached, the end of the LOB file is reached, or the number of blocks has been reached.

After filling in the values, press [F2] or the "Save" button to execute.

Part VI

6 Error Messages

Each page from this Help chapter will cover an area of R:Scope where an error may occur.

6.1 Starting R:Scope or Connecting a Database

Unable to connect this database
If R:Scope prompts you with a question when you are trying to connect a database and you answer No, you will not be able to connect.

Unable to open the file
You will get the message "Unable to open the file DBNAME.RX1" when someone else is using the database that you are trying to connect and also has MULTI set off. Alternatively, you are using the (Other) option and the database you specified does not exist where you tried to connect it.

You will also get this message for the DBNAME.RX3, DBNAME.RX2, or DBNAME.RX4 files if they are missing. R:Scope will not connect the database unless all the files are there.

Error reading the special character settings
Error reading the table listing block
Error reading the column listing block
-FATAL ERROR- Unable to read the internal structure block
The disk on which the database is stored has a problem. Use a utility program to check your disk.

Incorrect owner name
When connecting a database with R:Scope you are required to enter the correct owner name. If you incorrectly typed the owner name, try connecting again.

Incorrect owner password
If a password is defined for the database owner identifier, you are required to enter it before connecting to the database. If you incorrectly typed the owner password, try connecting again.

The database structure file is smaller than it should be. Do you wish to expand it and fill the missing information with 0's?
The structure file (DBNAME.RX1) is damaged. Answering Yes will allow you to connect the database. The damage might be so severe that repairing the database is not possible. The last block of information stored in the structure file is the Column block. (See the Database Info option in the Display menu.) You should print the table listings of your database (use the LIST ALL command at the R> prompt) before attempting to repair the error in this situation.

The values for this database's internal structure block are invalid. Do you wish to open this database with default values?
The DB Info block in the structure file (DBNAME.RX1) is severely damaged. Answering Yes will allow you to connect the database. The damage might be so severe that it is not possible to repair the database. The default start lengths for the blocks in the structure file are used. (See the Database Info option in the Display menu.)

6.2 Checking Database Info

Checking version flag ...
The flag that identifies the version of R:BASE that this database was created with should be between -1034 and -1001. This flag is currently set to: <value>
Change the version flag to -1001.
Version Flag Information

Checking structure file block offsets and lengths ...
The offset of block is set to <value1>. It normally is set to <value2>. The length of block is set to <value1>. It normally is set to <value2>
Blocknum is the number of the position of a given block in the DB Info section of the structure file. R:Scope starts numbering the blocks at 0. For example, if you choose the Database Info option from the Display menu, the block titled SET is the block referenced as block number 1. *Value1* is the value currently stored in the Database Info block of the structure file. *Value2* is the value R:Scope is expecting.

The offsets (block start) and block lengths for blocks 0 and 1 should be the same for all databases. If the number of table, column, or index entries is incorrect, you will get block offset errors.

If the results of the LIST ALL command in R:BASE at the R> prompt looks correct, use the UNLOAD or BACKUP commands to recreate the database. Refer to your R:BASE manuals for more information.

If the results of the LIST ALL command in R:BASE at the R> prompt does not look correct, try changing the block start and lengths to the expected values. Repairing the database might not be possible.

Checking data file length ...
The actual size of the data file is <value1> bytes.
This structure file expects it to be between <value2> and <value3> bytes.
Value1 is the actual size of the data file (DBNAME.RX2). You see this number when you list files with the R:BASE DIR command. *Value2* is the number stored in the DB Info block of the structure file minus 8192 bytes. *Value3* is the number stored in the structure file minus 1 byte. Use the Structure option in the Fix menu to change the number stored in the DB Info block to *value1* plus 1 byte in the DB Info block.

Checking index file length ...
The actual size of the index file is <value1> bytes.
This structure file expects it to be between <value2> and <value3> bytes.
Value1 is the actual size of the index (key) file (DBNAME.RX3). You see this number when you list files with the R:BASE DIR command. *Value2* is the number stored in the Database Info block of the structure file minus 512 bytes. *Value3* is the number stored in the structure file plus 4096 bytes. Use the Structure option in the Fix menu to change the number stored in the Database Info block.

Checking LOB file length...
The actual size of the LOB file (File 4) is <value1> bytes. The database structure file expects it to be between <value2> and <value3> bytes.
Value1 is the actual size of the LOB file (DBNAME.RX4). You see this number when you list files with the R:BASE DIR command. *Value2* is the next block number. *Value3* is the file size stored in the LOB file. Use the Structure option in the Fix menu to change the number.

The actual number of blocks in the free block list is <value1> blocks. The database structure expects <value2> blocks.
Value1 is found by walking the free block chain in File 4 and counting the blocks. *Value2* is the number stored in the LOB file. Use the Structure option in the Fix menu to change the number.

The actual offset to the end of the free block list is <value1>. The database structure expects it to be <value2>.
Value1 is the address of the last block found by walking the free block chain in File 4. *Value2* is the address stored in the LOB file header. Use the Structure option in the Fix menu to change the address. You can also use either the PACK or RELOAD commands in R:BASE to correct the error.

6.3 Checking Structure

This is a view.
With R:BASE, the row width for a view is 0. R:BASE does not store any information in the Column block in the structure file for a view. The number of rows is stored as -1.

This is an attached dBASE file
For an attached dBASE file, the number of rows is stored as -2 in the Table block of the structure file.

If you receive this message and the table is not supposed to be a dBASE file, the row count information for the table is incorrect. Use the Structure option in the Fix menu to change the number of rows for the table to a number greater than 0, and then use the Check Data option to see how many rows should be in the table.

This is an attached SERVER table
For an attached SERVER table, the number of rows is stored as -3 in the Table block of the structure file.

This is an attached SYSTEM view
For an attached SYSTEM view, the number of rows is stored as -4 in the Table block of the structure file.

The width in 2-byte words of each row for this table is set to <value> in the structure file. It should be within the range from 2 to 2100.
The row size for this table is outside the valid range. The minimum size of a table is 4 bytes, which is 2 words. The maximum row size for a table without NOTE columns is 4096 bytes (2048 words).

The width in 2-byte words of each row for this table is set to <value1> in the structure file. The sum of the lengths for all the columns in this table is <value2> words.
Value1 is the row width from the Table block of the structure file. R:Scope calculates *value2* from the Column block in the structure file. If a table does not contain any NOTE data type columns, *value2* is the sum of the storage size of the columns. Refer to Displaying Column Structure for information about the storage size of columns and data types.

If the table does contain NOTE data type columns, you will get this error message only if *value1* is smaller than the sum of the storage size of the columns that are not NOTEs, plus 4 bytes per NOTE. Refer to Displaying Column Structure for information about the storage size of columns and data types.

The number of columns for this table is set to <value>. It should be within the range from 1 to 400.
The number of columns for any table cannot be less than 1 or greater than 400. The number stored in the Table block of the structure file is outside this range.

The assumed number of rows for this table is set to <value>. It should be greater than or equal to zero.
The number of rows for a table cannot be less than 0. The number stored in the Table block of the structure file is negative and less than -4.

The pointer to the first row of data for this table is <value1>. It should be within the range from 0 to <value2>.
Value2 is the File 2 size stored in the Database Info block of the structure file. *Valuel* is less than 0 or greater than the size of File 2.

You might have an error in File 2 size in Database Info. You should have used Database Info in the Check menu to verify and correct this information before checking the structure.

The pointer to the last row of data for this table is <value1>. It should be within the range from <value2> to <value3>.
Value2 is the starting pointer stored in the Table block of the structure file. *Value3* is the File 2 size stored in the Database Info block of the structure file.

You might have an error in File 2 size in Database Info. You should have used Database Info in the Check menu to verify and correct this information before checking the structure.

The offset into the column list for the first column in this table is set to <value1>. The previous table's first column offset plus the number of columns in the previous table add up to <value2>.
R:Scope will perform this check only when checking all tables. *Value1* is the first column location as stored in the Table block of the structure file. *Value2* is the location R:Scope expected to find based on calculating the location from the number of columns in each table.

The offset in 2-byte words of this column into a row of data is set to <value1>. The previous column's offset plus the previous column's length in words adds up to <value2>.
Value1 is the word offset for the column as stored in the Column block of the structure file. *Value2* is the offset R:Scope expected to find by calculating the offset from the other columns in the table.

R:Scope calculates the values as follows: The offset for the fIrst column in a table is 1. The offset for the second column is 1 plus the storage size of column 1 (in words). The offset for the third column is the offset of the second column plus the storage size of column 2 (in words), and so on. Refer to Displaying Column Structure for information about the storage size of columns and data types.

The data type of this column is set to <value>. It should be within the range from 1 to 14.
Valid data types must be in this range. See Display Column Structure for a description of valid data type numbers. Fix structure will display both the number and the data type description. Computed columns will be displayed as the negative of the data type number, but must be in the range from -14 to-1.

The length for this text column* is *set to <value>. It should be within the range from* 1 *to 1500
The data type for this column is text (3) and the length stored in the Column block of the structure file is outside this range. Column lengths for all other data types should be 1.

6.4 Checking Data

ERROR* at *address <value>
This message is displayed above each of the following messages. *Value* identifies the address in the data file at which the error occurred.

The size* of *this row* of *data is set* to *<value1>.
The structure file expects the size* to *be* at *least <value2>.
The row size is less than expected. If a table contains no NOTE data type columns, the row size for each row in the table should be the same. *Value2* in this case would be from the Table block in the structure file. This number in words can be displayed using the Table Structure option in the Display menu. To get the correct number in bytes, multiply it by 2.

If a table does contain NOTE data type columns, value2 is the sum of the storage size of the columns that are not NOTEs, plus 4 bytes per NOTE. Refer to for information about the storage size of columns and data types.

The size* of *this row* of *data is set* to *<value1>.
The structure file expects the size* to *be* at *most <value2>.
The row size is more than expected. If a table does not contain NOTE data type columns, the row size for each row in the table should be the same. *Value2* in this case would be from the Table block in the structure file. This number in words can be displayed using the Table Structure option in the Display menu. To get the correct number in bytes, multiply it by 2.

If a table does contain NOTE data type columns, *value2* is 4200.

The pointer* to *the previous row* of *data in this table is set* to *<value1>. It should be within the range from 0* to *<value2>.
Value2 will be the address of the current row. You will get this error message when *value1* is larger than *value2.* If you get this error message, you will also get the next message.

The pointer* to *the previous row* of *data in this table is set* to *<value1>, but the address* of *the last row encountered is <value2>.
If you are using the Data option in the Fix menu to display the rows for the table, *Value2* is the address of the previous row displayed. It is the row whose next row pointer was the current address. Edit the row pointers from the Fix: Data screen to fix the problem.

The pointer* to *the next row* of *data in this table is set* to *<value1>. It should be within the range from <value2>* to *<value3>.
Value2 is the address of the current row. *Value3* is File 2 size from the Database Info block in the structure file. You can use several techniques to fmd the correct next row pointer value-see Manually Fixing the Data File.

Changes made to* a *note column in this table have caused this row to be relocated lower in the table. The location* of *the new row is <value1> and it should be within the range from <value2> to <value3>.
R:BASE changes the row size to 0 at the original address of the row when it moves a row with a NOTE field. If R:Scope sees a row size of 0, it assumes the row contains a NOTE column and that the row is a moved row. If you get this message and your table does not contain any NOTE fields, try changing the row size to match the row size stored in the Table block in the structure file. Use the Table Structure option in the Display menu to see the row size in words. Multiply the value by 2 to get the row size in bytes.

Value2 is the current row address. *Value3* is the ending pointer for the table.

The following messages apply to the table and not to a specific address:

User interruption

You will receive this message if you pressed a key to abort the check while R:Scope was checking the table data. Other error messages that follow might not indicate an actual error in the table.

Unable to read from the data file
The disk on which the database is stored has a problem. Use a utility program to check your disk.

The last address read for this table was <value1>. The structure file expects the end of table to be at <value2>.
Value2 is the ending pointer for the table from the Table block of the structure file. Usually this error is from a bad next row pointer, from a 0 next row pointer, because File 2 size is incorrect, or because the structure file information is out of sync with the data file information.

If the error is a problem with a next row pointer, you can use several techniques to find the correct next row pointer value-see Manually Fixing the Data File.

The number of rows counted for this table was <value1>. The structure file expects to count <value2> rows.
Value1 is the actual number of rows counted by following the next row pointers. *Value2* is the number of rows stored in the Table block of the structure file. This error might be corrected by fixing other errors.

LOB address <value1> is invalid.
Value1 is the pointer to the LOB file that is stored for a data row in the data file. The pointer does not point to a valid address in the LOB file.

The column pointer contained in LOB value <value1> is inconsistent (row address <value2>. column <value3> of table <value4> with the actual row.
A row of data is pointing to an address in the LOB file and that LOB address points back to a different File 2 address. *Value1* is the address in the LOB file. *Value2* is the address of a row in the data file that is stored in the LOB file. *Value3* is the column number stored for the LOB data and *value4* is the table number. You can fix the error by exporting the LOB data and attaching it to the correct row using R:BASE.

The LOB column value <value1> has an actual length of <value2> bytes. The data row expects if to be <value3> bytes.
The actual length of the data stored in the LOB file is different than the length stored in the data row in File 2. *Value1* is the address of the data in the LOB file. *Value2* is the actual length of the data in the LOB file. *Value3* is the length stored in the data file. Fix this error by using the RELOAD command in R:BASE.

The LOB column value address <value1> is beyond the end of LOB data file (File 4) at <value2>.
The LOB column value exceeds the actual file length of the LOB file. It is recommended that you RELOAD the database.

6.5 Checking Indexes

Below are possible errors when checking indexes:

Index rows counted: <value1>, expected count: <value2>

The index points to an invalid row location

The index points to a row that has been deleted

The index cannot be examined

If any of the errors are encountered when checking indexes, a PACK or RELOAD in R:BASE should resolve the issue.

6.6 Automatic Fix

No data exists for this table
You attempted to automatically fix data for a table whose starting row pointer is 0 or whose number of rows is 0.

This table contains a NOTE column and cannot be used with automatic fix.
Use the Manual option in the Fix menu to correct errors in this table.

There are too many errors present for an automatic fix to correct.
You cannot use the automatic fix option if R:Scope finds more than 100 errors when checking the data file for this table. Use the Manual option in the Fix menu to correct errors in this table.

<value> row(s) of data will be forfeited. Do you wish to continue?
If you answer No, the table is left unchanged. Use the Manual option in the Fix menu to correct errors in this table. If you answer Yes, R:Scope will update both the structure and data file for this table and report the changes on the Check Screen.

6.7 Fixing Data Manually

No data exists for this table.
You attempted to fix data manually for a table whose starting row pointer is 0 or whose number of rows is 0.

You cannot skip past the end of the data file
If you are using the [Tab] key to skip rows from the Fix: Data screen, the current address plus the current row size plus 10 is beyond the end of the data file. See Database Info in the Display menu for the size of File 2. Remember to use the Database Info option in the Check menu to verify that File 2 size is correct before fixing data.

Addresses must be between 1 and <value>
Value is the File 2 size from the Database Info block of the structure file. The address you specified when pressing [F7] to jump to a location in the data file is beyond the end of the data file. See Database Info in the Display menu for the size of File 2. Remember to use the Database Info option in the Check menu to verify that File 2 size is correct before fixing data.

Error reading the data file
The address R:Scope is trying to read is not in the data file. Generally you also have an error in File 2 size in Database Info. The File 2 size stored in the structure file is larger than the actual size of the data file. Remember to use the Database Info option in the Check menu to verify that File 2 size is correct before fixing data.

Another reason for this message is a problem with the disk on which the database is stored. Use a utility program to check your disk.

The address for the previous row is invalid
The previous row pointer is an address that is beyond the File 2 size stored in the structure file or is less than 0.

The address for the next row is invalid
The next row pointer is an address that is beyond the File 2 size stored in the structure file or is less than 0. The address might be a correct address. Remember to use the Database Info option in the Check menu to verify that File 2 size is correct before fixing data.

The current row is the top of this table The previous row pointer is 0.

The current row is the bottom of this table The next row pointer is 0.

Rows are not contiguous
When you are moving between rows on the Fix: Data screen with CONTIGUOUS set on, the next row pointer is not equal to the current row's address plus 10 plus the current row's size. R:Scope displays this

message and then when a key is pressed, displays the next row. The last row in a table will always be flagged as not contiguous.

Row not found
Search [F10] did not find a row based on the current search setting. Search will look through all of the data file; it does not stop searching at the File 2 size stored in the structure file if that number is smaller than the actual file size.

6.8 Chart

Unable to continue
The most likely reason is that a row size is invalid. The problem would be after the ending address for the last section of data that R:Scope read. This information is displayed after this error message is presented. You might want to check your data to see if any of your tables report a problem.

Part VII

7 Troubleshooting FAQ

Q. *What can I do to make my database easier to fix?*
A. One good idea is to reload your database in R:BASE periodically-every week, for example. A reloaded database has all the rows for a given table together in the data file until new rows are added or existing rows are changed. (An existing row is moved to a new location in the data file if there is a NOTE field in the table whose value has been edited so that the new value is larger than the old value.) A database can be reloaded by using either the RELOAD command (the Pack To option) or the BACKUP and RESTORE commands.

In addition, if you have tables with unique identifiers, such as an ID number, you might want to recreate the table in sorted order before reloading the database. This helps if you have a break in your database and you can't find some of the rows. It makes it easier to identify which rows you have lost.

Another factor is whether your tables have NOTE data type columns. Tables with NOTE columns do not have a fixed row length. If you have broken pointers, especially in more than one table, it is harder to figure out with which table to associate the rows. Also, you cannot easily correct an invalid row size when the table has NOTE columns.

Tables without NOTE columns have the same length for each row in the table. If you have broken pointers, especially in more than one table, it is easier to figure out which table to associate rows with when each table has a fixed row length. However, multiple tables with the same structure or the same row length make it difficult to figure out which rows belong to which table.

You should keep some hard copy documentation for your database to assist you in fixing the database. A listing of the forms, reports, rules, and labels, as well as the structure for each table in the database (LIST ALL from the R> prompt) are all useful. If you are using SQL passwords, you should also have a printout of the LIST ACCESS information.

Q. *I know that reloading my database makes all the rows in my tables contiguous. Is there any way I can tell when my tables are fragmented and that I should reload my database?*
A. Run the AUTOCHK command with the FULL option in R:BASE to check database integrity and to report the percentage of unused space in the data file. You can use the Chart option in R:Scope to see how fragmented the tables in your database are. Because the Chart option groups data by contiguous rows, the more groups you see for tables, the more fragmented the database is.

Q. *R:Scope doesn't display all the messages for my database in the Check Screen window.*
A. Check the settings under "Options > Line Count in Check Screen". You may have to increase the line count. For large databases, set the number to 10000.

Q. *R:Scope lets me look at and edit just about anything in the structure or data file. Are there some things that I shouldn't change using R:Scope or that would be easier to change using R:BASE?*
A. Yes, there are things that are easier to repair using R:BASE as well as areas of the files that can cause problems if you try to change them using R:Scope.

For example, you can use R:Scope to edit column information and change the data type of a column. However, this will not convert the data in the column to the correct data type and you might be unable to access the data in the column. Data types of columns should always be changed in R:BASE, not in R:Scope. Also, the database settings are much easier to change in R:BASE than in R:Scope. You can use R:Scope to delete rows from a table, but your indexes are not updated, so it is easier and usually faster to do this in R:BASE. In general, R:Scope should be used to repair damage to the structure and data files, not to do database maintenance or data and structure modification operations.

Q. *I see some table names I don't recognize when R:Scope displays the menu of table names in my database. I see names like sys-passwords and sys_computed. What are these tables?*
A. In addition to the system tables that are included in your database table listing (for example, *sysforms, sysrules, sysreports),* R:BASE has some hidden system tables that you won't see unless you use a utility like R:Scope to look at your database.

R:BASE uses the hidden *sys_computed* table to store information about computed columns, and the hidden *sys-passwords* to store access rights information. All the system tables and columns begin with *sys_.*

Q. *Why does R:Scope display my system tables? I always thought I wasn't supposed to modify anything in these tables.*
A. There might be invalid pointers in the system tables as well as in your data tables. If you have a bad pointer in one of your system tables, R:BASE will not be able to find all the information it needs from the system tables. You need to be able to repair damage to the system tables.

Q. *I used R:Scope to check my database, and some of the errors reported were in system tables. What happens if I can't recover some of the rows in my system tables?*
A. It is very important to keep documentation of information that R:BASE stores in the system tables.

A listing of the forms, reports, rules, and labels, as well as the structure for each table in the database (LIST ALL from the R> prompt) are all useful. If you are using SQL passwords, you should also have a printout of the LIST ACCESS information.

If your forms, reports, rules, or labels table are damaged and you are not able to recover all the rows in R:Scope, you might or might not be able to use a particular form, report, or label in R:BASE. Using part of the product that accesses a rule might give an error message.

For example, if your reports table is damaged, after recovering as many of the rows as possible in R:Scope, you should try printing each report from R:BASE to see which ones work and which ones might have lost part of their definition. If a particular report doesn't work, try modifying the report. If you can modify the report, you can probably fix it. If you can't modify the report, you will have to delete it and either recreate it or transfer the report from a backup copy of your database.

If the *sys_computed* table is damaged, after recovering as many of the rows as possible in R:Scope, you might need to redefine some of your computed columns in R:BASE.

Q. *I set REPEAT on to speed up moving through the rows in my table. All of a sudden, the rows on the screen dimmed, even though R:Scope continues to move through the rows, and I didn't receive an error message. Why?*
A. The rows on the screen probably dimmed because R:Scope encountered a bad previous row pointer. As you were moving down in the table and using the next row pointers, R:Scope did not give you an error message, but dimmed the rows to indicate that you could not use the previous pointers to return to these rows. When rows on the screen are dimmed, it means that the previous row pointers will not be able to chain back to these rows.

Q. *I don't understand why there is an option to turn off the display of rows in the Fix: Data screen. Why wouldn't you want to see the rows you're fixing?*
A. When you are using the next row pointers to move through a large number of rows (REPEAT on, [Down Arrow]), R:Scope will be able to move faster if it does not have to display the rows on the screen. R:Scope will stop and display a message when it reaches the end of the table or
when it encounters an invalid next row pointer. You can then turn the display on to look at the data. If you have CONTIGUOUS on, R:Scope will also stop with a message when it reaches a non-contiguous row.

Q. *When editing my pointer on the Fix: Data screen, I can't get R:Scope to accept my changes. The only way I can get out of editing is to press [Esc] and that isn't saving the changes.*
A. When you press [Esc] to exit editing pointers, R:Scope does not save your changes. You need to press [F2] after making your changes so that R:Scope will accept your changes and return you to the Fix: Data screen. Press [F2] to accept changes and continue. Press [Esc] to abort and discard changes.

Q. *I used R:Scope to undelete some rows that were accidentally deleted from a table, but I still can't access the rows.*
A. When R:Scope undeletes rows, it does not update the index file. The index will still show those rows as deleted rows until you rebuild the indexes in R:BASE. So, when you try to access the row using an indexed WHERE clause, R:BASE can't find the rows because the index indicates that the row is deleted.

Q. *I am trying to press [F9] from the Fix: Data screen, and it's not working. Why?*
A. If you choose All from the table list when choosing to fix the data field, some of the options are not available. The status line does not display a table name at this point. R:Scope needs to know the structure of the table to interpret the data when [F9] is pressed.

Q. *The numbers don't come out right. I'm in the Fix: Data screen, and I'm trying to compute the next row pointer by adding the row size to the current address. Why don't I get the right rows? I'm sure they are there.*

A. You probably forgot to include the 10 byte overhead per row in your calculation. To find the next location in the data file, you need to take the current address, add the row size for the current row, and then add the row overhead-10 bytes.

Q. *I relinked my rows, but when I press [F9] to look at the data on the reconnected rows, it's not right. How can I use R:Scope to correct the data on these rows?*
A. You cannot edit the actual data in your tables using R:Scope. R:Scope only lets you view the data in your tables. You'll need to use R:BASE to edit the data in these rows. It's often better to use R:BASE to delete the rows where the data is not displayed correctly and then reenter the rows.

Q. *I'm editing my table and column information and the changes I make aren't being saved.*
A. Press [F2] to accept changes and return to the menu. Press [Esc] to abort and discard changes. If you press [Esc] to leave a dialog box, such as the ones in which you edit column and table information, your changes are discarded. You must press [F2] to leave the dialog box and save changes.

Q. *I am editing some of my database information and I made a mistake. Is it possible to reset the value back to the original value, or get out of what I am doing without making the change?*
A. If you are in a dialog box (for example, choosing the Structure option on the Fix menu to change your table or column information), until you leave the dialog box by pressing [F2] to save your changes, you can press [Esc] to exit and not save your changes. Press [F5] to reset a single value in a dialog box until you move to the next field in the dialog box.

When you are editing values outside of a dialog box (for example, editing pointers on the Fix: Data screen), as soon as you move off of a field, the values are changed. Until you move out of the field you can press [F5] to reset the value or press [Esc] to exit the field and not save the change.

Q. *I want to add a database check option to my application. How can I call R:Scope from my application and get it to automatically start up and check my database?*
A. You can call R:Scope from your application using the ZIP command, but there is no way to have R:Scope automatically check the database. Use the AUTOCHK command in R:BASE.

Q. *When I'm using the Structure and Data options in the Check menu in R:Scope, sometimes the white bar doesn't go all the way to the end, and sometimes it just seems to be a solid white bar as it goes through the tables. Why?*
A. When checking the structure, it might happen so fast that the white bar appears solid. When checking the data, there are two reasons why the white bar might not go to the end. There might be a problem in the table's data and R:Scope was unable to read through all the data it expected for a table. Or, the data check happened so quickly that you were unable to see the bar go all the way to the end.

The white bar is simply meant to indicate how long the check will take if you have a large table. The Check Screen information tells you the exact status of what R:Scope found.

Q. *My computer is hanging in R:Scope. What could be causing this?*
A. The database might be so badly damaged that when R:Scope tries to interpret the information stored in the database, it causes R:Scope to hang.

Try using one of the sample databases included with the product. If R:Scope hangs with the sample databases, there might be a problem with your installation of R:Scope or an incompatibility between R:Scope and your computer, or your computer's configuration.

To pinpoint the problem, if you normally run on a network, reboot your computer as a single user.

If R:Scope still hangs after you follow these suggestions, install R:Scope on a different computer.

Q. *I have some enhancement suggestions for R:Scope. What is the best way to let R:BASE Technologies, Inc. know what they are?*
A. Refer to the Feedback section of this help.

Q. *Is there a way to print only part of the information on the Check Screen?*
A. You can clear the Check Screen by reconnecting the database. Choose only the options whose output you want to print. The check information is stored in the file ERROR.LOG. You can exit R:Scope and edit this file in a text editor, then print it.

Part VIII

8 Technical Support

Please read over the help documentation at least once before seeking support. We have worked very hard to make the help documentation clear and useful, but concise. It is suggested that you reread these instructions once you have become accustomed to using the software, as new uses will become apparent.

If you have further questions, and cannot find the answers in the documentation, you can obtain information from the below sources:

- Email our Technical Support Staff at: support@rbase.com
- Access the R:BASE Technologies Support home page online at https://www.rbase.com/support

You may be required to purchase a technical support plan. Several support plans are available to suit the needs of all users. Available Technical Support Plans

Please be prepared to provide the following:

- The product registration number, which is located on the invoice/order slip for the purchased product
- The type of operating system and hardware in use
- Details regarding your operating environment; such as available memory, disk space, your version of R:BASE, local area network, special drivers, related database structures, application files, and other files that are used or accessed by your application

All provide information will be used to better assist you.

R:BASE Technologies has a number of different services available for R:BASE products. As a registered user, you will receive information about new features for R:BASE and other R:BASE Technologies products. Please remember to register your software. https://www.rbase.com/register/

Part

9 Useful Resources

. R:BASE Home Page:	https://www.rbase.com
. Up-to-Date R:BASE Updates:	https://www.rbaseupdates.com
. Current Product Details and Documentation:	https://www.rbase.com/rbg11
. Support Home Page:	https://www.rbase.com/support
. Product Registration:	https://www.rbase.com/register
. Official R:BASE Facebook Page:	https://www.facebook.com/rbase
. Sample Applications:	https://www.razzak.com/sampleapplications
. Technical Documents (From the Edge):	https://www.razzak.com/fte
. Education and Training:	https://www.rbase.com/training
. Product News:	https://www.rbase.com/news
. Upcoming Events:	https://www.rbase.com/events
. R:BASE Online Help Manual:	https://www.rbase.com/support/rsyntax
. Form Properties Documentation:	https://www.rbase.com/support/FormProperties.pdf
. R:BASE Beginners Tutorial:	https://www.rbase.com/support/rtutorial
. R:BASE Solutions (Vertical Market Applications):	https://www.rbase.com/products/rbasesolutions

Part X

10 Feedback

Suggestions and Enhancement Requests:

From time to time, everyone comes up with an idea for something they'd like a software product to do differently.

If you come across an idea that you think might make a nice enhancement, your input is always welcome.

Please submit your suggestion and/or enhancement request to the R:BASE Developers' Corner Crew (R:DCC) and describe what you think might make an ideal enhancement. In R:BASE, the R:DCC Client is fully integrated to communicate with the R:BASE development team. From the main menu bar, choose "Help" > "R:DCC Client". If you do not have a login profile, select "New User" to create one.

If you have a sample you wish to provide, have the files prepared within a zip archive prior to initiating the request. You will be prompted to upload any attachments during the submission process.

Unless additional information is needed, you will not receive a direct response. You can periodically check the status of your submitted enhancement request.

If you are experiencing any difficulties with the R:DCC Client, please send an e-mail to rdcc@rbase.com.

Reporting Bugs:

If you experience something you think might be a bug, please report it to the R:BASE Developers' Corner Crew. In R:BASE, the R:DCC Client is fully integrated to communicate with the R:BASE development team. From the main menu bar, choose "Help" > "R:DCC Client". If you do not have a login profile, select "New User" to create one.

You will need to describe:

- What you did, what happened, and what you expected to happen
- The product version and build
- Any error message displayed
- The operating system in use
- Anything else you think might be relevant

If you have a sample you wish to provide, have the files prepared within a zip archive prior to initiating the bug report. You will be prompted to upload any attachments during the submission process.

Unless additional information is needed, you will not receive a direct response. You can periodically check the status of your submitted bug.

If you are experiencing any difficulties with the R:DCC Client, please send an e-mail to rdcc@rbase.com.

Index

- N -

- O -

- P -

- R -

- S -

- T -

- U -

- V -

Back Cover