R:BASE
Multi-User Guide

R:BASE Multi-User Guide

by R:BASE Technologies, Inc.

In a multi-user environment, R:BASE allows users to simultaneously access, view, update, insert, and delete data. To most efficiently use R:BASE on a network, R:BASE uses various types of settings, locks, and a waiting period to avoid situations that cause R:BASE to work more at preventing user conflicts than at storing, sorting, and retrieving data.

R:BASE Multi-User Guide

Printed: April 2025 in Murrysville, PA

First Edition

Table of Contents

Part I

1 Multi-User Considerations

When used within a multi-user network environment, R:BASE allows multiple users to simultaneously access, view, update, insert, and delete data from common database resources.

To ensure data integrity when multiple users attempt simultaneous access these resources, R:BASE applies a flexible set of locking and access mechanisms, termed "concurrency controls", to minimize contention between user requests. Several of these controls are automatically applied by the R:BASE engine, others are optionally applied by the developer. All possess default values, and each is configurable, individually, and in collaboration with other controls.

Through skillful application of these controls, the R:BASE developer has the opportunity to optimize the engine's management of concurrency, which in turn significantly enhances the engine's responsiveness to users' simultaneous requests for service.

The first chapter focuses exclusively on the application of concurrency controls to R:BASE multi-user database development. The second chapter covers the R:BASE installation and preparing the application for network use. Following chapters deal with optimization techniques applied within application code. Please bear in mind that these tools and techniques address only aspect of the optimization challenge—concurrent access—and that there are other considerations beyond the scope of this Guide:

- Know your hardware. Have the best servers, workstations, switches, cabling and wireless nodes you can afford. Ensure that each is competently specified, configured, installed and maintained by someone who knows his or her craft, as well as the history of your system.
- Know your cabled Ethernet network. Same.
- Know your wireless Ethernet network. Same. But also, know when a Remote Desktop approach can overcome the performance limitations of wireless networks, especially within manufacturing environments where interference and signal degradation may degrade transmission of data.
- When designing your database schema, take full advantage of the relational capabilities of R:BASE for the linking and manipulation of data stored in multiple tables.
- Learn and apply the rules of normalization to your schema so that data is as "atomized" as possible, thus minimizing the resources required by any user at a given moment.
- Take full advantage of temporary tables and views in your coding to reduce the number and frequency of (otherwise avoidable) "hits" on permanent tables.

Topics

Part II

2 Multi-User Concurrency Settings

R:BASE has always offered an excellent multi-user interface built upon flexible database and table-locking schemes, including concurrency control for data entry and editing. The concurrency control, locking, and resource waiting essentially prevents two users from modifying the same data at the same time. Concurrency control, row, table, and database locks only affect operations in which users are storing new data or altering existing data. R:BASE allows as much access to database resources as possible and keeps the duration of locking as brief as possible. However, the extent to which users are locked out of resources depends on the operations that cause R:BASE to issue the locks.

Concurrency Control
The most effective feature for data entry and editing is concurrency control. This automatic feature allows any number of users, using either forms or the EDIT command, to enter and edit data at the same time in one table or in many tables. Concurrency control takes advantage of the computer's speed to check whether the data user1 is manually modifying has been changed by someone else since user1 selected it. This leaves the control of change to the data in the hands of the last person who edits it. Concurrency control gives access of a particular row of data to several users, not limiting access to a table or a row of data to one user at a time. When the row of data is finally saved to the table, R:BASE does a quick table lock. This is to prevent other users from making structural changes.

Resource Waiting
When a command has been locked out of a resource--database, table, column--R:BASE checks to see if the requested resource is available while in the waiting period. If the resource becomes available, the user can proceed. If not, R:BASE displays a message informing the user that the resource is unavailable, ignores the command, and goes on to the next command.

When establishing the database and application in a multi-user environment, there are concurrency setting considerations to make. The following are the R:BASE concurrency settings, their values, and how the values can be applied. Each setting name can be selected to review their affect upon the application in a multi-user environment. You may need to change multiple settings to have the desired effect.

Setting	MULTI	STATICDB	FASTLOCK	PAGELOCK	ROWLOCKS	VERIFY	WAIT	INTERVAL
Code Location	Before CONNECT			Before or after CONNECT				
Default	ON	OFF	OFF	ON	ON	COLUMN	4	5
Value Options	ON|OFF	ON|OFF	ON|OFF	ON|OFF	ON|OFF	ROW| COLUMN	0-16383 seconds	0-9 tenths second
Required for all	YES	YES	NO*	NO	NO	NO	NO	NO
Optional for all	NO	NO	YES*	YES	YES	YES	YES	YES
Requires prior setting of		MULTI ON	STATICDB ON					
Enabled before CONNECT	YES	YES	YES	NO	NO	NO	NO	NO
Optionally made before/after CONNECT	NO	NO	NO	YES	YES	YES	YES	YES
Same value for all users	YES	YES	YES	NO	NO	NO	NO	NO
Different value for users	NO	NO	NO	YES	YES	YES	YES	YES
Set once per user	YES	YES	YES	NO	NO	NO	NO	NO
May change per user	NO	NO	NO	YES	YES	YES	YES	YES

* While ideal in multi-user operations, FASTLOCK is not required. However, if one user is connected to a database with FASTLOCK ON, all must do so as well.

The settings may be set within the R:BASE configuration file. However, the settings are best set in a startup file before connecting to the database. For example:

```
DISCONNECT
SET MULTI ON
SET STATICDB ON
SET FASTLOCK ON
SET PAGELOCK OFF
SET ROWLOCKS ON
SET VERIFY COLUMN
SET WAIT 4
SET INTERVAL 5
CONNECT ConComp
```

Related Topics:

Schema Reading Mode with SET STATICDB
Using SET ROWLOCKS to Lock Rows
Effects of the SET WAIT Command
Control Row Locks with SET PAGELOCK
Reduce Data Conflicts between Users with SET VERIFY
Table Locks
Other Multi-User Considerations

2.1 MULTI

Operating Condition

Syntax: SET MULTI ON/OFF

Default: OFF

SET MULTI sets multi-user capability on or off when you next connect a database. This setting must be used while you are disconnected from a database.

To set R:BASE so that it starts in multi-user mode by default, edit the R:BASE configuration file to include the command SET MULTI ON, or include the command in the startup file.

2.2 STATICDB

Syntax: SET STATICDB ON/OFF

Default: OFF

SET STATICDB activates a read-only schema mode. A user who first connects to a database with STATICDB set to on engages that database to operate in a read-only schema mode whereby any user must have their STATICDB setting on in order to connect to that database.

The effect of having STATICDB set on is that no schema changes can occur during that connection session.

Schema Reading Mode with SET STATICDB
In a multi-user environment, R:BASE must be aware of any schema modifications made by a user. Being aware of schema modifications, which is necessary for database integrity, entails constantly re-reading schema information.

Because many data-processing operations do not make frequent schema changes to the database, such a re-reading unnecessarily slows down processing.

The command SET STATICDB ON instructs R:BASE to prevent any schema modifications, therefore, R:BASE does not read schema information. By default, STATICDB is set off. SET STATICDB OFF forces a re-read of schema information before running any command.

When connecting to a database with STATICDB set on, R:BASE displays the "Database Schema is Read-Only." message. When STATICDB is set on, any changes made to add new tables or views results in a temporary table/view. The table/view will be dropped when disconnecting from the database. The LIST TABLE command will display the temporary table and view names with a "(T)" next to the name.

Also, the following R:BASE commands are not allowed when STATICDB is set on.

ALTER TABLE*	DROP*
COMMENT ON*	RBDEFINE*
CREATE INDEX	RESTORE

* Access is not allowed to tables created prior to database connection.

No ALTER, DROP, CREATE, SCONNECT, SATTACH, SDETACH, ATTACH, and DETACH commands will have any effect. You can issue CREATE, ATTACH, and SATTACH commands but they will only create temporary tables or views. Similarly, PROJECT will create temporary tables with or without the TEMP keyword. Because the database schema is protected from changes, R:BASE doesn't need to worry that a user will change the schema in the middle of a series of commands.

The UNLOAD ALL command does not act upon temporary tables. You can, however, unload individual temporary tables with the UNLOAD command.

If a user has STATICDB set off, that user cannot connect to a database being used by a user who has STATICDB set on. All users accessing the same database must have a matching STATICDB setting.

You can find out what the current STATICDB setting is with the SHOW STATICDB command. You can also capture the current STATICDB setting as a value with the CVAL function:

```
SET VAR vcval = (CVAL('STATICDB'))
```

2.3 FASTLOCK

Syntax: SET FASTLOCK ON/OFF

Mode: Multi-user and STATICDB

Default: OFF

Set FASTLOCK on for faster multi-user performance while modifying data, allowing the use of a quicker locking mechanism. With FASTLOCK on, R:BASE does not place a table lock on the table, allowing for greater throughput. A table lock is only needed to prevent structure changes.

FASTLOCK can only be set on when STATICDB is set on, and both FASTLOCK and STATICDB must be set on before the database is connected. Like other R:BASE database modes (SET MULTI and SET STATICDB), FASTLOCK requires all users to be connected with the same setting.

The following command lines set STATICDB and FASTLOCK correctly.

```
SET STATICDB ON
SET FASTLOCK ON
CONNECT concomp
```

2.4 PAGELOCK

Syntax: SET PAGELOCK ON/OFF

Mode: Multi-user

Default: ON

PAGELOCK specifies how R:BASE locks data when updating and deleting rows.

The settings for PAGELOCK are:

- **ON -** R:BASE uses page locking or row locking as appropriate. When PAGELOCK is ON and two or more users are updating rows within the same page of data, R:BASE only lets the first user update rows--the other users are locked out until the first user's update has been completed.

- **OFF -** R:BASE uses a fast row-locking method where only row locking is applied with no page locking. When PAGELOCK is OFF, you can lock rows of data instead of locking a page of data. You increase multi-user performance when PAGELOCK is OFF. And even more so when STATICDB and FASTLOCK are on.

Control Row Locks with SET PAGELOCK
If you know that your application mainly updates or deletes data a row at a time, rather than many rows, set PAGELOCK to OFF for row locking. In this case, R:BASE locks a row, reads the row, makes the change, and then releases the row. Otherwise, set PAGELOCK ON for page locking when you are doing an UPDATE and/or DELETE affecting many rows in a table.

Keep in mind that the PAGELOCK setting can be changed dynamically and can be different for different users using the same database.

Technically, the efficient and fastest method for updating data in multi-user environment is to SET STATICDB ON, SET FASTLOCK ON, and SET PAGELOCK OFF. This particular combination will result in the fewest contentions between users.

Notes:

- FASTLOCK and PAGELOCK can be set on at the same time.
- Setting STATICDB and FASTLOCK to ON (in that order), with PAGELOCK set to OFF will significantly increase multi-user performance with individual row changes.
- PAGELOCK is not the same as SET ROWLOCKS.
- Setting the value of PAGELOCK does not change the setting of ROWLOCKS.
- The PAGELOCK setting can be changed dynamically and can be different for different users using the same database.

Example:

```
-- The UPDATE must alter the values for may rows
SET FEEDBACK ON
SET PAGELOCK ON -- use page locking
UPDATE <tablename> SET <columnname> = value  -- no WHERE Clause
SET PAGELOCK OFF -- use row locking
SET FEEDBACK OFF
CLS
```

2.5 ROWLOCKS

Syntax: SET ROWLOCKS ON/OFF

Default: ON

R:BASE uses row-level locking in a multi-user environment. This command causes R:BASE to lock only the required row for the current command instead of locking the entire table. For example, if multiple users are modifying the same table using the UPDATE command, R:BASE locks only the rows affected by each UPDATE. When ROWLOCKS is set off, R:BASE sets table locks during each UPDATE, regardless of how many rows are affected.

Using SET ROWLOCKS to Lock Rows
When R:BASE is running in multi-user mode, the user has the added capability of forcing R:BASE to use row-level locking on some R:BASE commands. By default, the SET ROWLOCKS command is set on. Setting ROWLOCKS off is not recommended if all users intend to update the same tables. R:BASE always uses row-level locking for the EDIT, EDIT USING, and ENTER commands.

2.6 VERIFY

Syntax: SET VERIFY COLUMN/ROW

Default: COLUMN

SET VERIFY, used in the multi-user environment, specifies the level of concurrency control as a row or a column within a row.

SET VERIFY allows you to specify whether R:BASE concurrency control will apply to individual columns within a row or to all columns in the row. When the level of concurrency control is set to COLUMN, R:BASE checks only the columns you change while you are editing. When the level of concurrency control is set to ROW, if you change data in any column, R:BASE checks every column in the row.

Reduce Data Conflicts between Users with SET VERIFY
R:BASE concurrency control operates automatically when you are using a form in multi-user environments. Concurrency control includes autorefresh and verification. When you refresh or try to save a row, R:BASE checks whether data has been changed by another user and alerts you if it has changed. This prevents simultaneous changes that could corrupt the integrity of your data. The SET VERIFY command affects the operation of both autorefresh and verification when you are using a form.

When concurrency control is set to COLUMN, R:BASE looks for conflicts, those instances when two users have both modified the same column. When R:BASE detects a conflict in at least one field:

- R:BASE displays all of the other user's changes in the appropriate fields.
- Where there is no conflict, changes you made continue to be displayed.
- R:BASE displays a message informing you that data has changed.

When concurrency control is set to ROW, R:BASE looks for a change to any column in the whole row, whether it is a conflict or not. When R:BASE detects a change:

- R:BASE displays all of the other user's changes in the appropriate fields.
- Where there is no conflict, changes you made to the data are discarded.
- R:BASE displays a message informing you that data has changed.

Whether concurrency control is set to COLUMN or ROW, you can review the changes and then continue editing the data in the form. After autorefresh, R:BASE prompts you to press any key to continue editing. After verification, you can either move on to your next task or edit the data again. If you choose to move on when the level of concurrency control is set to COLUMN, you will be discarding any changes you made that are still displayed. R:BASE prompts you to press [Esc] if you want to move on, or to press [Enter] if you want to edit the displayed data.

When you edit data in a form, concurrency control is always enforced.

When you enter data in a form, concurrency control is enforced only when you are entering values in fields defined with a same-table look-up, or when you return to a row in a region that you had entered previously.

The first command line in the following example sets the level of concurrency control to check for changes in the entire row. The second command line starts an editing session using the form named custform.

```
SET VERIFY ROW
EDIT USING custform
```

2.7 WAIT

Syntax: SET WAIT value

Range: 0 to 16383 seconds

Default: 4

SET WAIT sets the minimum number of seconds to retry the last requested resource (a table or database) before halting execution. Rather than aborting execution, SET WAIT allows you to set a length of time for R:BASE to keep trying to access a resource.

If you do not run a SET WAIT command, R:BASE automatically retries the locked resource for approximately four seconds.

For commands that wait for resources, the precise period of the wait is at least as long as the time specified. On some computers, processing requirements may extend the length of the wait to longer than one second for each second designated.

When you enter a command from the R> prompt and the waiting period expires, R:BASE displays a message informing the user that the resource is unavailable. When the command runs as part of a command file, however, and the waiting period expires, R:BASE displays the unavailable resource message, ignores the command, and goes on to the next command.

The following command tells R:BASE to retry the last requested resource for approximately 20 seconds.

```
SET WAIT 20
```

You can also adjust the INTERVAL in which R:BASE tries the command during the SET WAIT period.

Effects of the SET WAIT Command
The SET WAIT command specifies the period in seconds in which R:BASE continues to retry the last command before stopping processing. If you set the wait period too long, you can have your users sitting unproductively for long periods. Be sure you set manual table locks (discussed later) only when absolutely necessary. If users are running programs that lock tables, set resource wait periods reasonably short to avoid users leaving workstations for a long without exiting the database.

A wait period can help prevent a deadlock. A deadlock occurs if a user locks a table and then waits for a table previously locked by a second user, who in turn is waiting for the table locked by the first user. R:BASE prevents deadlocks because a command either accesses a resource or eventually stops trying to obtain access.

2.8 INTERVAL

Syntax: SET INTERVAL value

Default: 5

Range: 0 to 9 tenths of a second

The SET INTERVAL command specifies the time to elapse before R:BASE retries the command that caused a conflict within the waiting period (see WAIT).

2.9 Table Locks

The below table describes the automatic concurrency control and locks used in multi-user mode and the commands that initiate them. Operations that only view data or the database structure are not affected by concurrency control or locking.

Type of Lock	Command	Description
Concurrency Control	EDIT EDIT USING ENTER	Prevents one user from accidentally overwriting another user's changes. These commands can access the same table simultaneously.
Row Lock	DELETE ROWS ENTER *form* INSERT *values* LOAD FROM *filespec*	When SET ROWLOCKS is on, a lock is only applied to a row. When off, a table lock is in effect.

	UPDATE	
Table Lock	BACKUP* DROP RULES FORMS GRANT INSERT *subselect* RBLABELS LOAD *from filespec* REPORTS RESTORE REVOKE RULES SET LOCK ON UNLOAD*	Must wait for commands that obtain table and database locks. Once obtained, this lock excludes all other concurrency control and locking until these commands have finished.
Full Database Lock	BACKUP ALL RELOAD UNLOAD ALL	All tables in the database are locked.
Database Schema Lock	ALTER TABLE CREATE SCHEMA CREATE TABLE CREATE VIEW CREATE INDEX DROP INDEX DROP COLUMN DROP TABLE DROP VIEW INTERSECT JOIN PROJECT RENAME SUBTRACT UNION	Engages a full database lock preventing schema modifications, then releases the schema lock remaining in table lock.
Cursor Lock	OPEN *cursorname*	Stops database schema commands but allows table locks; acts as a table lock.

*A table lock is placed only if one table is unloaded. A database lock is placed if more than one table is unloaded.

The below table shows what happens when different commands try to access a table or database already being used by another command. The columns represent the type of control or lock already placed on the table or database. Each row has a heading with the name of the control or lock needed by the command trying to gain access to a table or database.

	Concurrency Control	**Cursor Lock**	**Row Lock**	**Table Lock**	**Database Lock**	**Schema Lock**
Cursor Lock	Access	Access	Access	Wait	Quit	Wait
Row Lock	Access	Access	Access	Wait	Quit	Wait
Concurrency Control	Access	Access	Access	Wait	Quit	Wait
Table Lock	Wait	Wait	Wait	Wait	Quit	Wait
Database Lock	Wait	Wait	Wait	Wait	Quit	Wait
Schema Lock	Wait	Wait	Wait	Wait	Quit	Wait

2.9.1 Using SET LOCK to Set Exclusive Table Locks

The SET LOCK command sets locks on tables. SET Lock should be set on when a user wants to be certain that no other user will alter data in the tables being updated during a procedure. This command is useful in conjunction with the DECLARE CURSOR command. If R:BASE cannot lock all tables listed in the command, it does not lock any of the tables listed in the command.

Exclusive table locks are cumulative--that is, for each SET LOCK *tblname* on command you issue, you must issue a corresponding SET LOCK *tblname* off command to remove the lock from that table. Also, the user who locked the table must issue the SET LOCK *tblname* OFF command.

The SET LOCK command is typically used in an application program to set locks, allow a procedure to be performed, then remove locks.

Automatic locking is in effect even if the SET LOCK command is issued. Setting locks off affects only locks set by the SET LOCK *tblname* ON command--not locks that R:BASE sets automatically.

2.9.2 Displaying Multi-User Locks

When R:BASE is used on a network, the LIST command displays the names of any locked tables with a letter next to the table name for the type of multi-user lock.

The LIST TABLE command tells you whether the lock is an edit, cursor, row, local, or remote lock, as shown in the below table. The LIST TABLE LOCK command lists all locked tables within the database.

Letter	Lock	Description
r	Row lock	Another workstation is using EDIT ALL or a form that accesses this table or other commands that use row locks
L	Local lock	The SET LOCK command was issued from this workstation
R	Remote lock	A table lock has been applied with a command issued at another workstation
C	Cursor lock	A cursor is open on the table

2.10 Other Multi-User Considerations

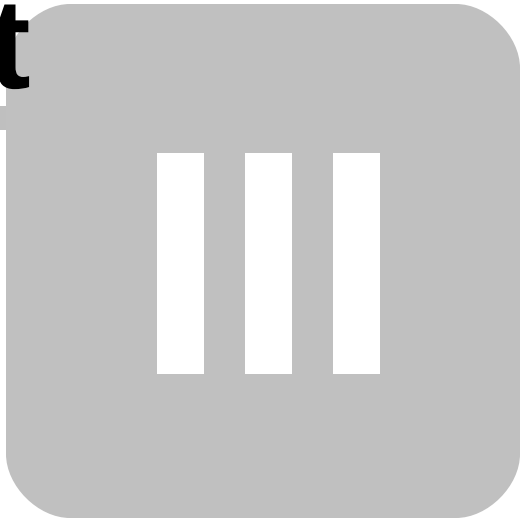
Only a limited number of users can access certain parts of R:BASE at the same time. Be sure that all network users are aware of the limitations and plan ahead to avoid conflicts in accessing database and application files. An administrator user should operate in single-user mode when performing any operation that modifies the structure of the database or using the PACK command.

The limitations to access for users are listed in the following table:

R:BASE Commands	Operating System Files Used	Access Limitations
CODELOCK	From ASCII to converted binary file	One user at a time can convert a given ASCII file
FORMS	Database files	One user at a time can modify a form in a given database.
REPORTS	Database files	One user at a time can modify a report in a given database.
GATEWAY	Database files	Data files for transfer to database, database files. One user at a time can use data files.
RBDEFINE	Database files	One user at a time can create or change database structure.
RBEDIT	ASCII file	One user at a time can create or edit a file.
LAUNCH	External programs	All users can share the program, within the limitations of the external program.

ZIP	External programs	All users can share the program, within the limitations of the external program.

Part



3 Preparing for Network Use

In a multi-user environment, you must ensure that the databases to be shared are located properly, that R:BASE is prepared to protect your data, and that all workstations are properly configured to share databases.

The database and custom application are stored on a shared network drive in order to allow multi-user access to the files. Each user must have the necessary read and write privileges in order to connect to the database. The network location must be accessible to all the workstations that will launch R:BASE and connect to the database. In addition to the security systems that most networks provide, the R:BASE GRANT command, with which access rights are assigned, provides additional security for R:BASE databases.

When R:BASE is on a local area network, users can share R:BASE program files, printers, hard disks, directories and databases. You can also have R:BASE program files on a local drive or workstation; however, those files will only be available to the user of that particular machine.

R:BASE on a network allows multiple users to simultaneously create and share new and existing databases, add, change, and print data, as if each user were the only one using that database. To ensure data integrity, R:BASE includes automatic concurrency control and a locking system that eliminates conflicts, such as two users attempting to change the same data at the same time. R:BASE has commands that allow a user to manually set the locks and wait periods that prevent conflicts.

Whether an R:BASE program or database files are stored on a network server or local hard drive, the limit on the number of seats using R:BASE remains in effect. Also, only a limited number of users can access some parts of R:BASE at the same time. For example, although users can share program files, only one user at a time can import data to a given database with the Import/Export utility, create or alter the structure of a database, or create, edit, or convert a given application file.

R:BASE can use most printers that are compatible with your network operating system. In some cases, you need only attach the printer to print from R:BASE. In other cases, you must identify the printer to the network operating system.

To set R:BASE up in a network environment, several steps need to be followed:

- Verify and secure how many software licenses (Per Seat for LAN or Remote Client for remote use) which will launch R:BASE
- Install R:BASE (client or server installation) for the workstations and/or servers
- Place all database and application files on a shared network drive that is accessible from all workstations
- Create or utilize an existing startup file to connect to the database and run the application, which is specified within a desktop shortcut

The following topics provide information you will need to when installing R:BASE on a multi-user system.

3.1 R:BASE Installation Options

In a multi-user environment, the best location for the R:BASE program files depend on three considerations:

1. Whether R:BASE Remote Client Licenses will be used for launching the application
2. The number of workstations where R:BASE will be launched, and subsequent frequency of R:BASE program files updates for those workstations
3. The age of server hardware versus the age of local workstations
4. The application is developed considering each user has their own R:BASE configuration file stored locally. The file may contain the user's name and is recognized in the menu system.

Remote Use
With Remote Client Licenses where users will connect to the server through a Remote Desktop (Terminal Services) connection, R:BASE must be installed on the server. Once R:BASE is installed, each remote user will use the defined desktop shortcut to open the application.

Workstation Count
Based on the number of workstations where R:BASE will be installed and launched, 10 or higher, and the availability of access to the workstations, it would be easier to install R:BASE on a shared network drive. Another benefit is that when applying R:BASE program updates, there is only one installation to be updated, rather than updating every workstation individually.

Comparing Hardware
In some cases, R:BASE may run faster from a local hard disk than from a network server. When program and application files are shared from a server, you can lose speed as users take turns reading the files and then wait while the files are transmitted over the network. With a newer faster server, this may not be the case. You would need to compare the workstation and server specifications to see which method is more effective. If R:BASE will be installed on the workstations, and the server will only be used to store the database and application, R:BASE does not need to be installed on the server.

For any workstation where R:BASE development will be actively performed, the R:BASE program should be installed on the local computer. This will allow access to the help manuals and documentation.

3.1.1 Client Installation

The Client Installation is the typical installation choice for single users or for users that want R:BASE installed upon the workstations, not the server, within a local area network. Using this method, you will run the installer which was made available through download while physically sitting at the workstation.

Make sure to log into the computer as the Administrator when installing R:BASE. Otherwise, you will not be able to install the software properly.

During the steps in the installation process, Full, Compact, and Custom installation options are available. These options will load the R:BASE executable into the program directory and the engine files into the system directory. For the "Custom" installation option, a different set of files can be installed for each "Components" option selected.

Components	Full	Compact	Custom	Server
Core Files	Yes	Yes	Yes	Yes
ODBC	Yes	Yes	Optional	Yes
Help Files (CHM)	Yes	No	Optional	No
Documentation (PDF)	Yes	No	Optional	No
Samples	Yes	No	Optional	No
Tutorial	Yes	No	Optional	No
R:BASE Editor Schemes	All	R:BASE Only	All	R:BASE Only

See Also Customizing the End User's Computer

3.1.2 Server Installation

The Server Installation is the typical installation choice for network administrators and developers that want R:BASE installed upon a centralized location within a local area network. It is beneficial to use this method when there are a great number of users which will be launching the R:BASE software. Another benefit is that when applying R:BASE updates, there is only one installation to be updated, rather than updating every workstation individually.

Using this method, you will run the installer ".exe", provided by download, while physically sitting at the server, or by remote access. Or, you can run the installer on a workstation and change the "Destination Folder" to a shared network directory during the Setup process, but this option prevents the proper installation of the R:BASE ODBC driver. If you have a R:BASE application which uses the R:BASE ODBC driver, you must run the installer at the server.

When running the installer, the "Components" screen will display a "Server Installation" option that must be selected.

After the installation is complete, users must be supplied with the necessary network access rights. The users will require read permissions to launch the R:BASE program. If you intend to store the database files in the same directory as the R:BASE program, then read and write permissions are required. For more information about these access rights, refer to the documentation for your network. Both mapped drive letters and universal naming conventions (UNC) are supported for the network shared directory.

Server Installation Considerations

- Default Settings - Each workstation that has their desktop set up appropriately with the shortcut properties will launch from the server correctly. However, all of the default user settings for R:BASE, which are loaded during the installation process, are now only located on the server's registry. This will leave the workstation's R:BASE Development interface with no stored settings forcing you or the user to set up the environment for the main Database Explorer, Form, Report and Label Designers, R:BASE Editor, R:Style, R> Prompt console, Data Browser, and Data Dictionary. In most cases a server installation means that end users will be running a custom application and will not need to access the development interface. If this is not the case, and you require that users have the ability to develop in R:BASE with a server installation, and would like the series of R:BASE default settings, a registry dump utility is available within the R:BASE program folder. Run the RBG11_Default_Settings.exe utility to load the registry with the default development environment settings. The utility may also be used to restore the default values if the registry settings become corrupted.

- (CVAL('NAME')) Function - Having the R:BASE configuration file stored in a single location means that all users will be recognized with the same name when using the (CVAL('NAME')) Function. An alternative is to use the Functions (CVAL('NETUSER')), which captures the logged in network user name, and/or (CVAL('COMPUTER')), which captures the actual computer name.

- Product Updates - After the server installation is implemented, you must remember that your method of applying R:BASE program updates has changed. When running an R:BASE update, be sure to use the "Server Update" button, which will drop all the program and DLL files into the specified directory.

- Launching the R:BASE Compiled Help (.CHM) - For those who are running the R:BASE development environment from a network installation and will launch the compiled help files (.CHM) from the network drive, the help files will not display properly. This is the result of a Microsoft security update which prevents users from running compiled Help files (.CHM) on network drives, as it may pose a threat. R:BASE Technologies, Inc. has no control over how the compiled help files are displayed, as this is an operating system update. Users may otherwise store and launch the R:BASE Help files on the local hard drives of the computers.

See Also Customizing the End User's Computer

3.2 Version Control

R:BASE must be used to develop the database and custom application. The R:BASE version and build used to create and customize the application must also be used to run the application. For example. if the users are running the custom application with R:BASE 11, the database and applications must have been developed with R:BASE 11. Along with the R:BASE product version, the product build should also match between the development environment and end user computers. For example, if the development computer is running R:BASE 11 Build: 11.0.1.20919, the end users should also run the exact build. To verify the version and build for an R:BASE installation, you can type SHOW VERSION at the R> Prompt.

```
R>SHOW VERSION
R:BASE 11, U.S. Version, Build: 11.0.1.20919
```

R:BASE Plugins
R:BASE plugins can be used to enhance, or extend R:BASE operations. R:BASE 9.x and higher versions use plugins with the .RBM file extension. R:BASE 7.x and Turbo V-8 versions use the .RBL file extension. The R:BASE plugin version must match the R:BASE software version in order to work properly.

3.3 Security Software Exceptions

Client and server computers may have one or more security-based software (anti-virus, anti-malware, anti-spyware) utilities installed.

Common to security programs is the ability to define "exceptions" for program files and folders where scans should be ignored.

Including R:BASE within the exception area is required in order to maintain high performance. If R:BASE itself is scanned each time it is launched, and the contents of the database and temporary files are halted and scanned each time they are created/accessed, users will see much slower response times when running R:BASE, making connections, and accessing files.

It is important to add exceptions to three areas of the R:BASE program:

Program Executable
For R:BASE 11, the executable name is RBG11.exe. The default installation folder is "C:\RBTI\RBG11".

For R:Compiler/Runtime for R:BASE applications, the compiled executable, Runtime executable, and DLLs (all of which are usually stored onto the shared network location) must be defined as exceptions.

Database Files
For R:BASE 11, the database files are the .RX1, .RX2, .RX3, and .RX4 files.

Although wildcard may be accepted, adding each file extension individually as an exception may be more appropriate.

Temporary Scratch Files
When R:BASE is launched, it will create three .$$$ temporary files. After performing certain actions, additional files may be created, and the existing files will increase and decrease in size. Add the *.$$$ files as an exception.

It is also recommended to scrutinize the security product, specifically how it handles exceptions!

Some security products treat exceptions differently, where scans (manual and scheduled) apply for "folder" exceptions, however real-time (application control) applies specifically to "file" exceptions.

The above could mean that all folder exceptions applied in a security product do not prevent the R:BASE executable from being scanned when launched, nor do the exceptions prevent the database files from being scanned while accessed!

3.4 Preparing the Application

It is common to distribute and store the custom application files in the same directory, ensuring your application will run correctly if it is ever moved. After installing the R:BASE program files, you need to install the database, application, startup, and configuration files that you created using R:BASE.

The following steps provide instruction for setting up R:BASE to run on a local area network.

1. Determine where to load the database, application and startup files (R:BASE application).

   Note: In a multi-user system, the database files should be located in a shared directory on the network server, not on a local drive.

2. Create the application directory on the network.
3. Copy the database, application, and startup files to the application directory.

   Note: For multi-user systems only, provide users with network read, write, and create access rights. For more information about these access rights, refer to the documentation for your network.

4. Copy the customized configuration file from the development computer to the program directory.
5. Determine where to load the R:BASE program files for the end users.

Note: These files can be located in a shared network directory on the network server or on each workstation local drive.

6. After installing and copying the files to the appropriate directories, store a backup of the database, application, startup, and configuration in a safe place.

The following lists the directory contents for the custom application on a multi-user system that contains only one application.

- Startup file (can include .DAT, .RMD, .RBA)
- Database files (.RX1-.RX4)
- Application files (can include .RFF, .RBA, .RMD, .EEP, .CMD, .APP, .etc.)
- Configuration file (.CFG)

The following lists other possible directory contents for a custom application.

- Plugin files (.RBM)
- R:Charts chart files (.RBC)
- R:BASE Themes DLL (RBThemes11.dll)
- R:BASE Gateway import/export specification files (.RGW)
- R:Synchronizer script files (.RSF)
- R:Mail Editor Template files (.RMT)
- Custom help file(s), if used in your application
- Moving GIFs, if referenced in your PAUSE/DIALOG commands

3.4.1 Temporary Scratch Files

Temporary files are created during R:BASE processing. The temporary files use the .$$$ file extension. Where they are stored can affect processing speed. In multi-user environments, it is very important that each user's temporary files be stored locally. The SCRATCH setting controls the directory location for the temporary files.

Syntax: SET SCRATCH <parameter>

Default: TMP

Parameter	Description
TMP	stores temporary files in the Windows TEMP directory
ON	stores temporary files in the database directory
OFF	stores temporary files in the current directory
<path>	stores temporary files in a specified directory

With SCRATCH set to ON, temporary files are stored on the database directory. This means slower operation in the typical multi-user environment because the database resides at the server, requiring the temporary files to be frequently transmitted over the network. For faster database speed, set SCRATCH to TMP. The TMP value directs R:BASE to store the temporary files on the current user's local "TEMP" directory, regardless of their operating system. This will help eliminate all issues related to the access rights, disk space, etc., when running R:BASE on enterprise servers. However, TMP requires your local workstation to have a hard disk with enough space to store the temporary files.

To set SCRATCH to TMP, use the following:

```
SET SCRATCH TMP
```

The SCRATCH command can be set in the configuration file so that the setting is made prior to launching R:BASE. Use the R:BASE Editor or any text editor to edit the configuration file to read the SCRATCH settings as follows:

```
SCRATCH TMP
```

3.4.2 Customizing the Configuration File

When R:BASE starts, the configuration file is used to set the default R:BASE operating environment including multi-user and environment settings, key map definitions and character tables. The configuration file sets the application environment.

Before changing a configuration file, make a backup copy of the file. Check your modifications carefully before saving the file, and test the edited file before you deliver the application to a user.

You can change the configuration file using the R:BASE Editor, or by using any ASCII text editor. The configuration file contains lines with semicolons as the first character, and lines with startup options. Insert lines beginning with a semicolon (;) to separate the file into sections or add comments. The changes you make will take effect the next time you start R:BASE or the custom application.

3.4.3 Creating the Startup File

A startup file is a command file that executes automatically when you launch R:BASE, such as CUST.DAT. A startup file for a custom R:BASE application ensures that users start the application the same way each time. The startup file is usually included in the same directory as the database and application files to start R:BASE, specify settings for the custom application, set up certain environment variables, enter passwords, create temporary tables, and/or close R:BASE when the application is closed.

A startup file can be an R:BASE command file (.DAT, .RMD, .CMD) or an R:BASE application file (.RBA).

Startup files can be specified in two ways:

1. **Desktop Shortcut Properties**

   Within the "Target:" field of a desktop icon's shortcut properties, you can specify a startup file. Specify your startup file in the "Target:" field after the R:BASE executable name. Then, separate the R:BASE executable and startup file with a space.

2. **R:BASE Configuration File**

   Inside the configuration file a startup file can be specified with the STARTUP parameter. This tells R:BASE to run the specified file when launched. To specify a startup file in the configuration file, you can edit the configuration file directly, or use the "Display" tab within the R:BASE Configuration Settings dialog. To access the Configuration Settings dialog, choose "Settings" > "Configuration Settings" from the main Menu Bar.

   To edit the configuration file directly, start the R:BASE Editor and open the configuration file.

   Before the OPTIONS parameter, insert the line STARTUP *filespec*. For example:

```
;============================================================
; Launch on Startup
;============================================================
MENU
STARTUP     C:\RBTI\RBG11\MainData\NewSet.dat
OPTIONS
```

Notes:

- If a startup file is specified in both the configuration file and the desktop shortcut, R:BASE executes the commands in the startup files specified in the configuration file first, then any file specified after the R:BASE executable.
- If an RBASE.DAT file is located in the "start in" or "working" directory upon launching R:BASE, the program will always automatically run this file.
- Do not include the following commands in a startup file:

- GATEWAY
- ZIP ROLLOUT

- If you are using CodeLocked procedure files (APP, APX) as the source for your R:BASE custom application, you would create an R:BASE startup file to initialize the procedure file. The startup file would only require two commands, RUN and EXIT:

```
RUN appname IN appname.apx
EXIT
```

Examples:

Example 01:

```
--CONCOMP.DAT
--ConComp Application Startup file using a form as the menu system
IF(CVAL('DATABASE')) <> 'CONCOMP' OR (CVAL('DATABASE')) IS NULL THEN
  CONNECT CONCOMP IDENTIFIED BY NONE
ENDIF
CLS
EDIT USING MenuForm
EXIT
```

Example 02:

```
--CONCOMP.DAT
--ConComp Application Startup file using codelocked files as the menu system
RUN CONCOMP IN CONCOMP.APX
EXIT
```

Example 03:
-- Automates the process of connecting to a database in a network environment with SET STATICDB ON, SET FASTLOCK ON, SET ROWLOCKS ON, and SET PAGELOCK OFF.

```
-- MyApp.DAT Startup Application File
-- Start Fresh
   CLEAR ALL VARIABLES
LABEL StartFresh
   DISCONNECT
   SET QUOTES=NULL
   SET QUOTES='
   SET DELIMIT=NULL
   SET DELIMIT=','
   SET LINEEND=NULL
   SET LINEEND='^'
   SET SEMI=NULL
   SET SEMI=';'
   SET PLUS=NULL
   SET PLUS='+'
   SET SINGLE=NULL
   SET SINGLE='_'
   SET MANY=NULL
   SET MANY='%'
   SET IDQUOTES=NULL
   SET IDQUOTES='`'
   SET CURRENCY '$' PREF 2 B
   DISCONNECT
   SET STATICDB ON
   SET ROWLOCKS ON
```

```
    SET FASTLOCK ON
    SET PAGELOCK OFF
    SET MESSAGES OFF
    SET ERROR MESSAGES OFF
    SET ERROR MESSAGE 2495 OFF
    CONNECT dbname IDENTIFIED BY ownername
    SET ERROR MESSAGE 2495 ON
    SET MESSAGES ON
    SET ERROR MESSAGES ON
-- Check the availability of database
IF SQLCODE = -7 THEN
    CLS
    PAUSE 2 USING 'Unable to Connect the Database.' +
    CAPTION ' Your Application Caption Here ...' +
    ICON WARNING +
    BUTTON 'Press any key to continue ...' +
    OPTION BACK_COLOR WHITE +
    |MESSAGE_FONT_NAME Tahoma +
    |MESSAGE_FONT_COLOR RED +
    |MESSAGE_FONT_SIZE 11
    CLOSEWINDOW
    EXIT
ENDIF
    -- Enforce Database Default Settings
    SET QUOTES='
    SET DELIMIT=','
    SET LINEEND='^'
    SET SEMI=';'
    SET PLUS='+'
    SET SINGLE='_'
    SET MANY='%'
    SET IDQUOTES='`'
    SET CURRENCY '$' PREF 2 B
    SET NULL ' '
    SET DATE FORMAT MM/DD/YYYY
    SET DATE SEQUENCE MMDDYY
    SET DATE YEAR 30
    SET DATE CENTURY 19
    CLS
    EDIT USING ApplicationMainMenu
    RETURN
-- End here ...
```

3.5 Customizing the End User's Computer

For each individual workstation, a desktop shortcut should be used to open the R:BASE program and launch the custom application on the server. If R:BASE is installed on the client workstation, the desktop shortcut should already exist. If R:BASE is installed on the server, the desktop shortcut must be created.

The following procedures are to be performed on the end user's computer, so an application automatically runs after double-clicking the desktop shortcut. Procedures for both client and server installations are provided.

Client Installation

1. Locate and right click on the R:BASE desktop shortcut.
2. Choose "Properties", then select the "Shortcut" tab.

Within the "Target:" field, the R:BASE executable and path should be listed.

3. At the end of the executable name, add a space, then add your startup file name. Then, add the "-BASE" command parameter if the instance is launching a multi-application application on a server.

   Example:

   ```
   C:\RBTI\RBG11\RBG11.EXE MainApp.dat -BASE
   ```

4. Under "Start in:", edit the path to read the network shared folder where the custom application and database files reside. To be sure of the path, you can navigate to the network location and copy/paste the path from the operating system address bar.

   Example:

   ```
   F:\RBDATA\MainApp
   ```

5. Select the "Apply" button.
6. Under the "General" tab, edit the shortcut name to whatever you choose.
7. Save your changes by selecting the "OK" button.
8. Double click the icon to launch the application.

Server Installation

1. From the end user workstation desktop, navigate to the network folder where the R:BASE program files (executable) were installed.
2. Right click on the R:BASE executable (e.g., RBG11.EXE), select "Send To" > "Desktop (create shortcut)".
3. Then, on the desktop, right click and select "Properties" for the new desktop icon.
4. From the "Shortcut" tab, add a space and then the "-a" parameter to the end of the "Target:" field value after the executable name. The executable and "-a" parameter must be separated with a space.
5. After the "-a" parameter, add a space, then add your startup file name. Then, add the "-BASE" command parameter for use in server environments.

   Example:

   ```
   R:\RBTI\RBG11\RBG11.EXE -a MainApp.dat -BASE
   ```

6. Select the "Apply" button.
7. Under the "General" tab, edit the shortcut name to whatever you choose.
8. Save your changes by selecting the "OK" button.
9. Double click the icon to launch the application.

3.5.1 Startup Parameter for Server Environments

It is advised to use the "-BASE" command line parameter to initiate optimized Database Explorer settings for use in active server environments. When implemented, the parameter allows for faster use of R:BASE for database administrators and users alike, by disabling many display and appearance features that are very likely hidden during application runtime.

The following setting changes are automatically performed when "-BASE" is specified:

- use of the "View As List" display will omit displaying column details for objects
- UINOTIF setting is turned OFF, where the Database Explorer will not refresh for object changes
- the Show Slave Table setting is disabled within the Database Explorer
- the Show View Tables setting is disabled within the Database Explorer
- the Indicator For Compressed Forms/Reports/Labels is disabled within the Database Explorer
- the Show Row Numbers is disabled within the Database Explorer
- the custom Explorer Appearance settings are disabled within the Database Explorer
- the product splash screen is not displayed

The above values are set/established on load, meaning that when R:BASE is closed, the "minimal" settings will be preserved.

The following is an example of using the "-BASE" parameter within an R:BASE desktop shortcut:

Target: C:\RBTI\RBG11\RBG11.EXE "-BASE"

Be sure to use the startup parameter to speed up the database CONNECT time in Multi-User LAN, WAN, and Cloud environments!

Part IV

4 Increasing Application Performance

Many factors, both large and small, can affect database performance. So, fine-tuning your code and database is essential. Below are several areas to optimize the performance of your application and database to increase productivity:

4.1 Index Efficiency

When applied properly, indexes decrease the time it takes to get data from the database. However, when indexes are used poorly, or not at all, it will take longer for R:BASE to return data. When properly implemented, indexes can greatly improve the data retrieval performance of your applications.

When adding an index to a column, consider the following criteria:

- Columns that are not primary, foreign, or unique keys, but are frequently referred to in queries and sorts
- Columns that have rules applied to them
- Linking columns in views
- The index is more efficient if each row contains a value
- The index is more efficient based on the uniqueness of the data. Check the Duplicate Factor and Adjacency Factor values within the Data Designer.

In order to maintain index efficiency, review the following drawbacks to avoid:

1. Indexes added to a column where data duplication is high

   The single most important factor in determining the effectiveness of an index is the uniqueness of index values. A unique index value is found faster than a value with multiple index occurrences.

2. Indexes added to an existing Primary Key, Foreign Key, Unique Key

   Primary Key, Foreign Key, and Unique Key are all type of constraints that are automatically indexed when added to a table. Avoid indexing these keys as this creates a double index, and makes R:BASE perform twice as much work to retrieve the same data.

3. Indexing column where NULL values exist

   An index is more efficient if each row contains a value. A column with many NULL records will slow down the index.

4. Lengthy text column indexes

   The fastest, most efficient data types for indexed access are INTEGER, REAL, DATE, TIME, and TEXT with a defined length of four characters or less. It advised that you avoid lengthy TEXT column indexes.

5. Common column searches that are not multi-column indexes

   If you consistently search three columns when working with a database, you can define a separate index for each column, but it is better to define a multi-column index for all three columns. This is because R:BASE searches a multi-column index faster than three separate indexes.

6. Ascending/Descending "ORDER"

Data retrieval can be increased in an ORDER BY clause when the column or columns listed in the ORDER BY clause, are included in an index, with the same column sort order, as specified in the ORDER BY clause.

For example, processing data by invoice date in a descending order would be faster as those dates are likely to appear first. So, when defining the index for invoice date, the index should be created in descending order as well.

7. Not using indexes where needed

Over time, the structure of a database is likely to change. Periodically, you should check for places where indexes can be added to improve speed.

For a more in-depth review of indexes, review the "Indexing Explained" technical document available at the From The Edge Web site.

4.2 R:BASE Themes

Themes are artistic representations which enhance the visual display of an R:BASE screen. There are 86 pre-defined themes available in R:BASE. Themes can be applied to Forms, External Forms, Applications, the Print Preview window and the CHOOSE, DIALOG, PAUSE, PRNSETUP and #WHERE dialog windows.

External themes can be loaded into R:BASE. External theme files must have the ".msstyles" file extension. The themes can be acquired from any online resource or from a custom theme you created yourself. After a theme is loaded into the R:BASE session, the theme name specified will be listed in every available location where themes are specified and will appear in the Data Dictionary list. When using external themes, the theme file must be loaded into R:BASE each time the session is launched.

In the scenario where the R:BASE installation is on the server, with the R:BASE Themes DLL also on the server, the end users may see extended processing times when using themes in the custom application. The reason is that each of the 86 themes are stored within the DLL file, which is approximately 90MB in size. So, for every form and/or custom dialog window that uses a theme, R:BASE must read the DLL from the server and transfer it across the network.

Processing times can be decreased for R:BASE server installations using themes by performing the following:

- load external themes into R:BASE, rather than using the native R:BASE themes
- removing themes from forms that contain a large number of controls, as a theme is applied to each control

In order to load a theme into R:BASE, the PROPERTY command must be used. The following PROPERTY command parameters are to manage external themes in R:BASE:

Parameter	Value 1	Value 2	Description
LOAD_THEME	theme name	theme file	loads a new theme for the session
RELEASE_THEME	theme name	TRUE	releases an existing loaded theme
CHANGE_THEME	theme name	new theme file	loads a new a different theme for a loaded theme name

Load Theme Example:

```
PROPERTY LOAD_THEME 'NewLuna' LunaBlue.msstyles
```

After this command is issued, the theme "NewLuna" will be available in all locations where a theme can be specified.

Release Theme Example:

```
PROPERTY RELEASE_THEME NewLuna TRUE
```

After this command is issued, the theme "NewLuna" will be released.

Change Theme Example:

```
PROPERTY CHANGE_THEME NewLuna LunaXP.msstyles
```

After this command is issued, the NewLuna theme will use the styles defined in the LunaXP.msstyles file. This command parameter is provided to alter the theme file, and to avoid changing the specified "theme name" in every location within the code.

4.3 Optimizing Command Syntax Techniques

Use the following techniques in your command files and EEPs to speed up processing:

- **Group similar commands**

  R:BASE loads into memory the information needed to process each command. If the command is already in memory, R:BASE does not need to read the disk again. For example, try grouping separate SET VARIABLE commands into one SET VARIABLE whenever possible.

  When UPDATE commands are used for the same table using the same WHERE Clause, the commands should also be combined.

  ```
  UPDATE Donors SET Contrib = .vAmount WHERE DonorID = .vDonorID
  UPDATE Donors SET Renew = .vRenew WHERE DonorID = .vDonorID
  UPDATE Donors SET GiftDate = .vDate WHERE DonorID = .vDonorID
  UPDATE Donors SET Gift = .vNewGift WHERE DonorID = .vDonorID
  ```

  The following command combines the four UPDATE commands into one.

  ```
  UPDATE Donors SET Contrib = .vAmount, Renew = .vRenew, GiftDate = .vDate, Gift =
  .vNewGift WHERE DonorID = .vDonorID
  ```

- **Minimize disk access**

  Minimize disk access by using Custom EEPs, which run from the computer's memory, and by combining small command files into command blocks within a procedure file.

- **Use WHILE loops**

  Use WHILE loops instead of IF structures or GOTO processing whenever possible. All the commands contained within a WHILE loop are completely read and are retained in memory as long as the WHILE loop is processing. Use BREAK or change the condition for a natural exit rather than using GOTO to exit from a WHILE loop.

- **Replace SET VARIABLE with SELECT INTO for Table Lookups**

  When assigning column values to one or more variables, it is better to use SELECT ... INTO rather than SET VARIABLE. SELECT INTO is the SQL compliant command when capturing table data into variables, and allows for specifying an INDICATOR variable which indicates if a column value is NULL.

  SET VARIABLE syntax:
  ```
  SET VARIABLE vPhone = EmpPhone IN Employee WHERE EmpLastName = 'Smith' AND
  EmpFirstName = 'George'
  ```

  SELECT Syntax:
  ```
  SELECT EmpPhone INTO vPhone INDICATOR ivPhone FROM Employee WHERE EmpLastName =
  'Smith' AND EmpFirstName = 'George'
  ```

- **Use forward GOTO searches**

Use forward GOTO searches whenever possible. When R:BASE encounters a GOTO, it performs a forward search for the matching label more efficiently than by seeking it in memory, even though the labels are retained internally.

- **Predefine variables**

  Make sure that the data type of each variable is unambiguous by explicitly assigning the data type when the variable is defined.

- **Reduce redundancy**

  Eliminate duplication of command sections where possible. If you have a set of commands duplicated in several places within a form, use a "Custom Form Action" which can be called upon at any place in the form. If you have a set of commands duplicated in several places within the entire application, use stored procedures, which can be called upon from almost any place within the application. Like Custom EEPs, stored procedures run from the computer's memory.

- **Eliminate rules checking**

  Set RULES OFF when not needed for data verification. This saves time by preventing R:BASE from checking the data for rule violations. It may also be possible to eliminate the rule with a constraint.

- **Experiment with Manual Table-Order Optimization**

  By default, R:BASE uses its own internal algorithm optimizer that determines the best order for joining tables. You can turn off the automatic optimizer using the MANOPT setting. MANOPT (default: OFF) disables the automatic table-order optimization that R:BASE performs when running queries. This gives maximum control over the order in which columns and tables are assembled in response to a query. With MANOPT set to ON, R:BASE uses the order of the tables in the FROM clause and the order of the columns in the column list of the SELECT clause to construct the query.

4.4 Finding Minimum and Maximum Values

This section demonstrates techniques for finding the minimum and maximum values from the same table.

Note the use of two separate COMPUTE commands. Example 2 runs faster because R:BASE allows both COMPUTEs to be executed in one command. Example 3 is the fastest because it uses the SELECT INTO *varlist* command.

```
SET VAR t1 TIME = NULL
SET VAR t2 TIME = NULL
SET VAR vDiff INTEGER = NULL

--Example 1 - Slow
SET VAR t1 = (.#TIME)
SET VAR vMin INTEGER
SET VAR vMax INTEGER
COMPUTE vMin AS MIN Altitude FROM Airports
COMPUTE vMax AS MAX Altitude FROM Airports
SET VAR t2 = (.#TIME)
SET VAR vDiff = (.t2 - .t1)
PAUSE 2 USING .vDiff

--Example 2 - Faster
SET VAR t1 = (.#TIME)
SET VAR vMin INTEGER
SET VAR vMax INTEGER
COMPUTE vMin AS MIN Altitude, vMax AS MAX Altitude +
  FROM Airports
```

```
SET VAR t2 = (.#TIME)
SET VAR vDiff = (.t2 - .t1)
SET VAR vTime = (RTIME(0,0,.vDiff))
PAUSE 2 USING .vTime

--Example 3 - Fastest
SET VAR t1 = (.#TIME)
SET VAR vMin INTEGER
SET VAR vMax INTEGER
SELECT MIN (Altitude), MAX (Altitude) INTO vMin, vMax +
  FROM Airports
SET VAR t2 = (.#TIME)
SET VAR vDiff = (.t2 - .t1)
SET VAR vTime = (RTIME(0,0,.vDiff))
PAUSE 2 USING .vTime
```

4.5 Surrounding the WHERE Clause with Parentheses

You can easily gain processing speed by NOT surrounding your WHERE clauses with parentheses where non-text values are being used. When R:BASE sees parentheses, the values are evaluated as text, even though the values are not. By removing the parentheses, your code speeds will increase.

```
SET VAR t1 TIME = NULL
SET VAR t2 TIME = NULL
SET VAR vDiff INTEGER = NULL
SET VAR v1 TEXT = 'Seattle'
SET VAR v2 INTEGER = 93
SET VAR v3 INTEGER = 1000

--Example 4 - Slow
SET VAR t1 = (.#TIME)
SET VAR CheckNum INTEGER = 0
WHILE CheckNum < 40 THEN
SELECT AptCode INTO vAptCode INDICATOR ivAptCode +
FROM Airports WHERE (StCode = .v2 AND CityName +
CONTAINS .v1 AND Runway > .v3)
SET VAR CheckNum = (.CheckNum + 1)
ENDWHILE
SET VAR t2 = (.#TIME)
SET VAR vDiff = (.t2 - .t1)
SET VAR vTime = (RTIME(0,0,.vDiff))
PAUSE 2 USING .vTime

--Example 5 - Faster
SET VAR t1 = (.#TIME)
SET VAR CheckNum INTEGER = 0
WHILE CheckNum < 40 THEN
SELECT AptCode INTO vAptCode INDICATOR ivAptCode +
FROM Airports WHERE StCode = .v2 AND CityName +
CONTAINS .v1 AND Runway > .v3
SET VAR CheckNum = (.CheckNum + 1)
ENDWHILE
SET VAR t2 = (.#TIME)
SET VAR vDiff = (.t2 - .t1)
SET VAR vTime = (RTIME(0,0,.vDiff))
PAUSE 2 USING .vTime
```

4.6 Accumulating Data with SELECT

With R:BASE's procedural language, you can use various code structures to accumulate variables from the data values in two different tables. Example 6 uses two DECLARE CURSOR commands to repeatedly locate the values and do the accumulations until all data has been used. Example 7 is faster. It uses one DECLARE CURSOR and one COMPUTE command to accumulate the sum. Example 8, the fastest, uses an INSERT command with a SELECT statement to accumulate the sum.

```
SET VAR t1 TIME = NULL
SET VAR t2 TIME = NULL
SET VAR vDiff INTEGER = NULL

--Example 6 - Slow
SET VAR t1 = (.#TIME)
SET VAR vLength = 0

DROP TABLE Totals
CREATE TABLE Totals (State TEXT 15, Length INTEGER)
DECLARE c1 CURSOR FOR SELECT State, StCode +
FROM States +
WHERE State IN ('Washington', 'Oregon', 'California')
OPEN c1
WHILE SQLCODE = 0 THEN
FETCH c1 INTO vState INDICATOR ivState, +
  vStCode INDICATOR ivStCode
IF SQLCODE <> 0 THEN
BREAK
ENDIF
DECLARE c2 CURSOR FOR SELECT Runway +
FROM Airports WHERE StCode = .vStCode
OPEN c2
WHILE SQLCODE = 0 THEN
FETCH c2 INTO vRunway INDICATOR ivRunway
IF SQLCODE = 0 THEN
SET VAR vLength = (.vLength + .vRunway)
ENDIF
ENDWHILE
INSERT INTO Totals VALUES (.vState, .vLength)
DROP CURSOR c2
ENDWHILE
DROP CURSOR c1
SET VAR t2 = (.#TIME)
SET VAR vDiff = (.t2 - .t1)
SET VAR vTime = (RTIME(0,0,.vDiff))
PAUSE 2 USING .vTime

--Example 7 - Faster
SET VAR t1 = (.#TIME)
DROP TABLE Totals
CREATE TABLE Totals (State TEXT 15, Length INTEGER)
DECLARE c1 CURSOR FOR SELECT State, StCode FROM States +
WHERE State IN ('Washington', 'Oregon', 'California')
OPEN c1
WHILE SQLCODE = 0 THEN
FETCH c1 INTO vState INDICATOR ivState, +
```

```
  vStCode INDICATOR ivStCode
IF SQLCODE <> 0 THEN
BREAK
ENDIF
COMPUTE vLength AS SUM Runway FROM Airports +
WHERE StCode = .vStCode
INSERT INTO Totals VALUES (.vState, .vLength)
ENDWHILE
DROP CURSOR c1
SET VAR t2 = (.#TIME)
SET VAR vDiff = (.t2 - .t1)
SET VAR vTime = (RTIME(0,0,.vDiff))
PAUSE 2 USING .vTime

--Example 8 - Fastest
SET VAR t1 = (.#TIME)
DROP TABLE Totals
CREATE TABLE Totals (State TEXT 15, Length INTEGER)
INSERT INTO Totals SELECT State, SUM(Runway) +
FROM States,Airports +
WHERE States.StCode = Airports.StCode +
AND State IN ('Washington', 'Oregon', 'California') +
GROUP BY State
SET VAR t2 = (.#TIME)
SET VAR vDiff = (.t2 - .t1)
SET VAR vTime = (RTIME(0,0,.vDiff))
PAUSE 2 USING .vTime
```

4.7 Using Nested Cursors

Nested cursors must be used correctly for efficient execution of code. Notice how the following examples use the DECLARE CURSOR command differently. Both examples work correctly, but only the second attains normal processing speed.

In Example 9, the DECLARE CURSOR is placed inside the WHILE loop of the first cursor. This requires the second DECLARE CURSOR command to execute every time the first command finds a row. In Example 10, both DECLARE CURSOR commands are executed at the start of the program, and the second DECLARE CURSOR command is opened inside the WHILE loop of the first command. By restricting the second DECLARE CURSOR from executing as often, this example runs much faster.

```
SET VAR t1 TIME = NULL
SET VAR t2 TIME = NULL
SET VAR vDiff INTEGER = NULL

--Example 9 - Slow
SET VAR t1 = (.#TIME)
DECLARE c1 CURSOR FOR SELECT State, StCode +
FROM States +
WHERE State IN ('Washington', 'Oregon', 'California')
OPEN c1
WHILE SQLCODE = 0 THEN
FETCH c1 INTO vState INDICATOR ivState, +
  vStCode INDICATOR ivStCode
IF SQLCODE <> 0 THEN
BREAK
ENDIF
```

```
DECLARE c2 CURSOR FOR SELECT Runway +
FROM Airports WHERE StCode = .vStCode
OPEN c2
SET VAR vLength = 0
WHILE SQLCODE = 0 THEN
FETCH c2 INTO vRunway INDICATOR ivRunway
IF SQLCODE = 0 THEN
IF vRunway > 5000 THEN
SET VAR vRunway = (.vRunway + 100)
ELSE
SET VAR vRunway = (.vRunway + 50)
ENDIF
UPDATE Airports SET Runway = (.vRunway) +
WHERE CURRENT OF c2
ENDIF
ENDWHILE
DROP CURSOR c2
ENDWHILE
DROP CURSOR c1
SET VAR t2 = (.#TIME)
SET VAR vDiff = (.t2 - .t1)
SET VAR vTime = (RTIME(0,0,.vDiff))
PAUSE 2 USING .vTime

--Example 10 - Faster
SET VAR t1 = (.#TIME)
SET VAR vRunway INTEGER = NULL, vStCode INTEGER = NULL
DECLARE cursl CURSOR FOR SELECT State, StCode +
FROM States +
WHERE State IN ('Washington', 'Oregon', 'California')
DECLARE c2 CURSOR FOR SELECT Runway FROM Airports +
WHERE StCode = .vStCode
OPEN c1
WHILE SQLCODE = 0 THEN
FETCH c1 INTO vState INDICATOR ivState, +
  vStCode INDICATOR ivStCode
IF SQLCODE <> 0 THEN
BREAK
ENDIF
OPEN c2
SET VAR vLength = 0
WHILE SQLCODE = 0 THEN
FETCH c2 INTO vRunway
IF SQLCODE = 0 THEN
IF vRunway > 5000 THEN
SET VAR vRunway = (.vRunway + 100)
ELSE
SET VAR vRunway = (.vRunway + 50)
ENDIF
UPDATE Airports SET Runway = (.vRunway) +
WHERE CURRENT OF c2
ENDIF
ENDWHILE
CLOSE c2
ENDWHILE
DROP CURSOR c1
```

```
SET VAR t2 = (.#TIME)
SET VAR vDiff = (.t2 - .t1)
SET VAR vTime = (RTIME(0,0,.vDiff))
PAUSE 2 USING .vTime
```

4.8 Displaying Messages in Long Tasks

Command Routines
It is helpful to show a progress message when performing long running tasks. When processing a time-consuming routine, there are ways to let the user know the status of the process using the PAUSE 3 with GAUGE options as well as the use of PROCESSMESSAGE command, which processes messages that are currently in the windows message queue. The PROCESSMESSAGE can help in the GUI part to avoid the "Not responding" behavior in Windows operating systems. A common use of PROCESSMESSAGE is in long WHILE loops.

PROCESSMESSAGE may be called in each loop iteration to give the GUI time to process the pending Windows messages. For a loop that only does data processing, PROCESSMESSAGE can also be used. It is advised to disable GUI update settings like UINOTIF before entering the loop with PROCESSMESSAGE, to counter some side-effects of PROCESSMESSAGE.

If is also important to not overuse PROCESSMESSAGE. Use the command only in places where it is necessary for the GUI to "breath" during a long running task.

PROCESSMESSAGE can be called after every iteration in the WHILE loop cycle. The following example demonstrates where to best place the PROCESSMESSAGE command.

When using the PAUSE, the NO_FOCUS option can also be used so the dialog will not be focused when displayed, which can be used to possibly prevent an interruption in the focus transition in an active form.

```
PAUSE 3 USING ' Calculating ... Please Stand By ...' +
CAPTION ' PAUSE Gauge with PROCESSMESSAGE' ICON APP +
OPTION GAUGE_VISIBLE ON +
|GAUGE_COLOR GREEN +
|GAUGE_INTERVAL 10 +
|MESSAGE_FONT_NAME Verdana +
|MESSAGE_FONT_SIZE 10 +
|MESSAGE_FONT_COLOR BLUE +
|NO_FOCUS
SET VAR vcounter INTEGER = 1
WHILE vcounter < 2500000 THEN
  SET VAR vcounter = (.vcounter + 1)
  PROCESSMESSAGE
ENDWHILE
CLEAR VARIABLE vcounter
CLS
```

Forms
Windows will flag a form as "not-responsive" if it is not able to process messages in the queue after some time. It is not advisable to use PAUSE as progress indicator for long running form tasks. A progress indicator "within" the form represented as a panel is preferred. A panel at the center of the form can be displayed, updated after every step in the routine, and then hidden after the routine completes. The logic below will assist in avoiding the "Not responding" Windows behavior in forms.

```
PROPERTY SaveButton ENABLED FALSE
PROPERTY ProgressPanel VISIBLE TRUE
--Step #1
PROPERTY ProgressPanel CAPTION 'Working on stuff #1...'
PROCESSMESSAGE
...
...
```

```
...
--Step #2
PROPERTY ProgressPanel CAPTION 'Working on stuff #2...'
PROCESSMESSAGE
...
WHILE ...
  ...
      ...
      --if a single iteration is quick enough, PROCESSMESSAGE can be called after a
few iterations
      PROCESSMESSAGE
ENDWHILE
...
--Step #3
PROPERTY ProgressPanel CAPTION 'Working on stuff #3...'
PROCESSMESSAGE
...
...
...
PROPERTY ProgressPanel VISIBLE FALSE
PROPERTY SaveButton ENABLED TRUE
```

Part V

5 Increasing Performance in Forms

This section illustrates multiple ways of improving form performance, by taking advantage of the latest improvements in R:BASE, and replacing older methods with quicker, more efficient results.

5.1 Moving Form Lookup Variables into Custom EEPs

When upgrading databases from older versions, it is important to take into account if and how R:BASE may work differently. Applications created with legacy logic can perhaps be improved upon in new releases.

For instance, in legacy versions of R:BASE, form variables were not calculated automatically. A RECALC VARIABLES command was needed in an EEP to refresh the variables and generate different results. In newer releases of R:BASE (7.0 and higher), form variables are recalculated automatically when the cursor moves from field to field.

Because of this logic change in R:BASE, the response time may be longer in forms with many lookup variables which are based upon large tables.

To retain that same performance when using the form, the lookup variables can be populated within an EEP.

In the following variables list, there are 11 lookups performed for the "Client" table based upon a provided client identification number. The variables were meant to display a range read-only information about the client.

```
Form :  OrderEntry
Main Table :  Orders
1  : TEXT   vClientFirstName = CFirstName IN Client WHERE ClientID = ClientID
2  : TEXT   vClientLastName = CLastName IN Client WHERE ClientID = ClientID
3  : TEXT   vClientCompany = CCompany IN Client WHERE ClientID = ClientID
4  : TEXT   vClientAddress1 = CAddress1 IN Client WHERE ClientID = ClientID
5  : TEXT   vClientAddress2 = CAddress2 IN Client WHERE ClientID = ClientID
6  : TEXT   vClientCity = CCity IN Client WHERE ClientID = ClientID
7  : TEXT   vClientState = CState IN Client WHERE ClientID = ClientID
8  : TEXT   vClientZipCode = CZipCode IN Client WHERE ClientID = ClientID
9  : TEXT   vClientPhone = CPhone IN Client WHERE ClientID = ClientID
10 : TEXT   vClientFax = CFax IN Client WHERE ClientID = ClientID
11 : TEXT   vClientEmail = CEmail IN Client WHERE ClientID = ClientID
```

Instead, the variable values can be populated using a SELECT command placed within an "On Exit" Custom EEP, within the control where the client ID was entered.

```
-- On Exit EEP
SET VAR vClientFirstName = NULL
SET VAR vClientLastName = NULL
SET VAR vClientCompany = NULL
SET VAR vClientAddress1 = NULL
SET VAR vClientAddress2 = NULL
SET VAR vClientCity = NULL
SET VAR vClientState = NULL
```

```
SET VAR vClientZipCode = NULL
SET VAR vClientPhone = NULL
SET VAR vClientFax = NULL
SET VAR vClientEmail = NULL
SELECT +
   CFirstName, +
   CLastName, +
   CCompany, +
   CAddress1, +
   CAddress2, +
   CCity, +
   CState, +
   CZipCode, +
   CPhone, +
   CFax, + +
   CEmail +
INTO +
   vClientFirstName INDIC iv1, +
   vClientLastName INDIC iv1, +
   vClientCompany INDIC iv1, +
   vClientAddress1 INDIC iv1, +
   vClientAddress2 INDIC iv1, +
   vClientCity INDIC iv1, +
   vClientState INDIC iv1, +
   vClientZipCode INDIC iv1, +
   vClientPhone INDIC iv1, +
   vClientFax INDIC iv1, +
   vClientEmail INDIC iv1 +
FROM Client WHERE ClientID = .vClientID
RECALC VARIABLES
RETURN
```

In the following, lookup data within a "ThirdParty" table, based upon a third-party payer ID, variables were also used to display a range read-only information.

```
12 : TEXT   vTPPContactName = TPPContactName IN ThirdParty WHERE TPP_ID = TPP_ID
13 : TEXT   vTPPCompany = TPPCompany IN ThirdParty WHERE TPP_ID = TPP_ID
14 : TEXT   vTPPConract = TPPContract IN ThirdParty WHERE TPP_ID = TPP_ID
15 : TEXT   vTPPCertificate = TPPCertificate IN ThirdParty WHERE TPP_ID = TPP_ID
16 : TEXT   vTPPNotes = TPPNotes IN ThirdParty WHERE TPP_ID = TPP_ID
```

These variable values can be populated using a similar SELECT command placed within an "On Exit" Custom EEP, within the control where the third-party payer ID was entered.

```
SET VAR vTPPContactName = NULL
SET VAR vTPPCompany = NULL
SET VAR vTPPConract = NULL
SET VAR vTPPCertificate = NULL
SET VAR vTPPNotes = NULL
SELECT +
   TPPContactName, +
   TPPCompany, +
   TPPContract, +
   TPPCertificate, +
   TPPNotes ,+
INTO +
   vTPPContactName INDIC iv1, +
   vTPPCompany INDIC iv1, +
```

```
      vTPPConract INDIC iv1, +
      vTPPCertificate INDIC iv1, +
      vTPPNotes INDIC iv1 +
   FROM ThirdParty WHERE TPP_ID = .vTPP_ID
   RECALC VARIABLES
   RETURN
```

Other variables which perform calculations must remain in the Form Variables. The variables will continue to be updated if any changes are made to the variables contained within the expressions.

```
17 : TEXT   vDayOfWeek = (SGET((TDWK(DateContarct)),3,1))
18 : CURRENCY   vSubTotal = (SUM(ExtPrice))
19 : CURRENCY   vFreight = (.vSubTotal * .01)
20 : CURRENCY   vSalesTax = (.vSubTotal * .07)
21 : CURRENCY   vInvoiceTotal = (.vSubTotal + .vFreight + .vSalesTax)
```

Note:

Using such technique, make sure to predefine all variables with appropriate data type as On Before Start EEP, and clear all necessary variables as On Close EEP.

```
-- Example
-- On Before Start EEP
   SET VAR vClientFirstName TEXT = NULL
   SET VAR vClientLastName TEXT = NULL
   SET VAR vClientCompany TEXT = NULL
   SET VAR vClientAddress1 TEXT = NULL
   SET VAR vClientAddress2 TEXT = NULL
   SET VAR vClientCity TEXT = NULL
   SET VAR vClientState TEXT = NULL
   SET VAR vClientZipCode TEXT = NULL
   SET VAR vClientPhone TEXT = NULL
   SET VAR vClientFax TEXT = NULL
   SET VAR vClientEmail TEXT = NULL
   SET VAR vTPPContactName TEXT = NULL
   SET VAR vTPPCompany TEXT = NULL
   SET VAR vTPPConract TEXT = NULL
   SET VAR vTPPCertificate TEXT = NULL
   SET VAR vTPPNotes TEXT = NULL
   RETURN

-- On Close EEP
   CLEAR VARIABLES iv%,vClient%,vTPP%
   RETURN
```

You will also need to create two variables as Form Expression to capture the values for entered ClientID and TPP_ID columns.

```
-- Example
nn : INTEGER   vClientID = (ClientID)
nn : INTEGER   vTPP_ID = (TPP_ID)
```

5.2 Command Syntax in EEPs

As more and more R:BASE users discover just how powerful Entry/Exit Procedures (EEPs) can be in forms processing, several methods have emerged that can help improve their performance.

Use Custom EEPs and Custom Form Actions

With the enhancement that all Custom Form Actions and Custom EEPs (Forms/Reports/Labels) are executed in memory instead of temporary files (version 7.6 and higher), another way to speed up the execution of EEPs is to move all the command file EEPs to Custom EEPs. This eliminates the need for R:BASE to create the temporary files to the disk when executing a command file EEP. You can make the transition to Custom EEPs by performing the following:

1. In the Form Designer, select one of the controls which references a command file EEP.
2. Right click on the control and select "Properties'.
3. From the "EEPs" tab, select the "Edit EEP..." button, which will display your EEP code in the R:BASE Editor.
4. Use the keyboard shortcut [Ctrl] + [A] to highlight the entire contents of the command file EEP.
5. Use the keyboard shortcut [Ctrl] + [C] to copy the contents to the Windows clipboard.
6. Close the R:BASE Editor window to return to the control properties dialog.
7. Select the "Edit Custom EEP..." button, which will open the R:BASE Editor to store the code, only this time the commands will be stored within the form itself.
8. Use the keyboard shortcut [Ctrl] + [V] to paste the entire contents of the command file EEP.
9. Select the "OK" button to return to the control properties dialog.
10. Remove the EEP file name from the "Custom" field so the EEP does not fire from the command and the Custom EEP, and only the Custom EEP.

The control properties dialog should now alter the color of the "Edit Custom EEP..." button to yellow.

Leave the EEP Quickly
When writing an EEP, decide in the first couple of lines whether the user is going to stay in the EEP or return to the form. For example, let's say you want the user to execute hundreds of lines of code if they press [F2] inside a field. Here is one way to accomplish this:

```
SET VAR vLAST = (LASTKEY(0))
IF vLAST = '[F2]' THEN
{HUNDREDS OF LINES OF CODE}
ENDIF
RETURN
```

This EEP will work, but even if the user doesn't press [F2], it requires the reading of hundreds of lines of code. The following structure is more efficient:

```
SET VAR vLAST = (LASTKEY(0))
IF vLAST <> '[F2]' THEN
   RETURN
ENDIF
   {HUNDREDS OF LINES OF CODE}
RETURN
```

This EEP is quicker because if the user doesn't press [F2], it evaluates only four lines of code. The bulk of the code is evaluated only if the user does press [F2]. So, when processing lots of conditional commands, use a GOTO command to jump to the end of the command file or use a RETURN command as soon as the process is completed.

Keep the User Informed
Speed is a matter of perception as much as a value to measure. In fact, the fastest EEP is only as fast as the user perceives it to be. Even if your EEP takes only 10 seconds or less to execute, by not informing the user that something is going on, you risk leaving the impression that something is wrong with the form.

To prevent user frustration, put a PAUSE command at the beginning of the EEP that lets the user know the EEP is being processed. Something like "Calculating ... Please Stand By ..." is probably sufficient. Example code for a PAUSE dialog with an oscillating gauge can be something like:

```
PAUSE 3 USING 'Calculating ... Please Stand By ...' +
CAPTION 'Calculating ...' ICON APP +
OPTION GAUGE_VISIBLE ON +
|GAUGE_COLOR [R218,G228,B246] +
|GAUGE_INTERVAL 10 +
```

```
|MESSAGE_FONT_NAME VERDANA +
|MESSAGE_FONT_SIZE 10 +
|MESSAGE_FONT_COLOR BLUE +
|THEMENAME Razzmatazz
 -- Put your code, that takes a lot of time, here ...
 -- Use CLS command to clear the PAUSE 3 dialog
CLS
RETURN
```

5.3 Disable RBTI Variable Processing

An added PROPERTY APPLICATION is available parameter to specify if "RBTI Variable" (RBTI_*) processing is enabled in forms. If perhaps the variables are seldom used in an application, the processing may be turned off, as the variables are assigned in nearly all forms aspects. The property is ON by default. To take advantage of the optimization, set the value to OFF upon the application startup. Then, turn it ON only in forms that use the RBTI_* variable values. Make sure the property is turned OFF again when the form is closed.

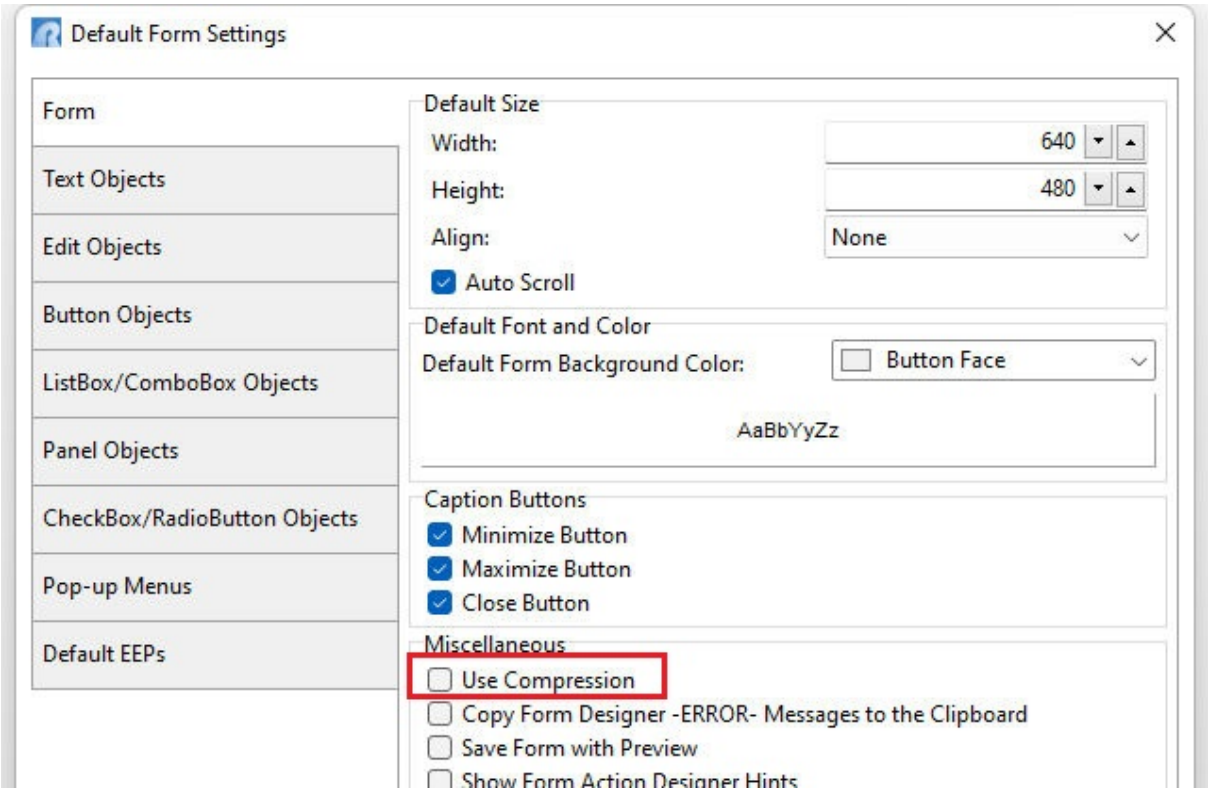
```
PROPERTY APPLICATION CAPTURE_RBTI_FORM_VARS OFF
```

5.4 R:BASE Form Compression

Form Compression is an available R:BASE setting which allows you to store all R:BASE forms as compressed within the large object data file (.RX4). The default value for Form Compression is set on. With Form Compression, you are taking advantage of smaller amounts of data being transferred across your network.

However, with Form Compression enabled, you cannot view or search the SYS_DATA column details within the in SYS_FORMS3 system table. One would be able to take advantage of this option for viewing the raw details of the form.

To view the setting, choose "Settings" > "Form Designer" from the main Menu Bar. Within the "Miscellaneous" panel, there is the "Use Compression" check box.



5.5 DB Grid and Enhanced DB Grid

In some cases, you can replace portion of your R:BASE form with a DB Grid or Enhanced DB Grid. With the addition of so many new enhancements for the Grid control, it is becoming one of the most widely used controls in R:BASE. Through these enhancements the DB Grid boasts more power for the developer to customize how the DB Grid will be used and to best suit the client's needs. Other enhancements have allowed the DB Grid to become more user-friendly for searching and editing records. And, with the latest visual enhancements, a Grid can be customized to display colorful and compelling data representations.

A. DB Grid "Power"

- In addition to all Form System Variables, the variable **RBTI_DBGRID_COLUMN** holds the name of focused "column" on DB Grid. This direct access to the control's focus allows you to see what is being selected.

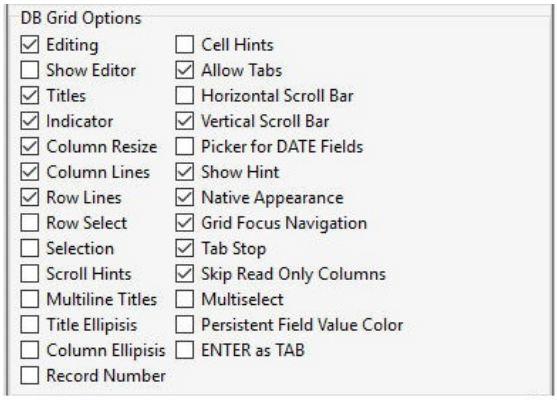
  You can also take advantage of the following RBTI Form Variables:

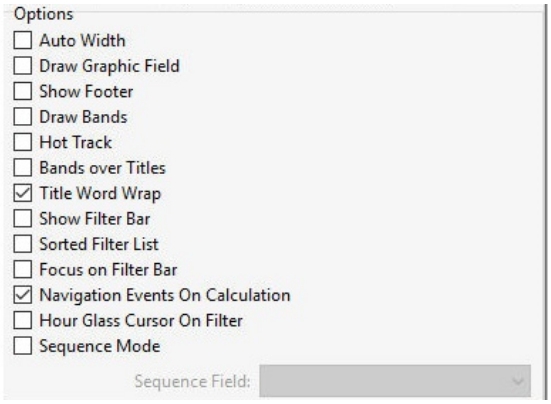
  - **RBTI_DIRTY_FLAG** - returns a 1 if any DB Control value(s) in the form is changed or 0 if nothing was changed
  - **RBTI_FORM_ALIAS** - holds the name of focused form, if used AS alias.
  - **RBTI_FORM_COLNAME** - holds the name of the focused column DB Control on the form
  - **RBTI_FORM_COLVALUE** - holds the value of the focused column DB Control on the form
  - **RBTI_FORM_COMPID** - holds the value of focused DB/Variable Edit control's Component ID, if defined
  - **RBTI_FORM_DATATYPE** - holds the data type of the focused column DB Control on the form
  - **RBTI_FORM_DIRTYVAR** - returns a 1 if any Variable Control value(s) in the form is changed or 0 if nothing was changed
  - **RBTI_FORM_FORMNAME** - holds the name of the current form. Particularly useful when using the form-in-a-form technique and multiple forms are running at the same time.
  - **RBTI_FORM_MODE** - holds the value of the current mode of the form, such as ENTER, EDIT or BROWSE

- **RBTI_FORM_TBLNAME** - holds the name of the current table in a form. This is especially useful when used within a multi-table form.
- **RBTI_FORM_VARNAME** - holds the name of the focused Variable Control on the form
- **RBTI_FORM_VARVALUE** - holds the value of the focused Variable Control on the form

- There are **fourteen** different EEPs available within Enhanced DB Grids.
  - On Entry into DB Grid
  - On Exit from DB Grid
  - On Entry into DB Grid Column
  - On Exit from DB Grid Column
  - On DB Grid Column Moved
  - On DB Grid Cell Click
  - On DB Grid After Cell Click
  - On Double Click
  - On Group Expression
  - On Column Widths Change
  - On Filter Change
  - On Mouse Enter
  - On Mouse Leave
  - On Top Left Change

- Within the Enhanced DB Grid Properties and Enhanced Options, the following can be set on or off:





B. DB Grid "User Friendliness"

- The Column Titles can be selected to sort the data in both ascending and descending orders.
- The Grid Columns can be customized with your own specified color, font, and alignment for both Title Properties and Column Properties. A "Read Only" setting is also available.
- A DATE picker calendar can be displayed for DATE data type fields.
- Cell hints can be added for additional end user instructions.
- Titles can be extended to display multiple lines.
- Title and column can be minimized with an ellipsis to suppress extra words.
- The row record number can be displayed within the left side of the Grid.

C. Grid "Visual Illustration"

- The Column Titles can be graphically enhanced with a multi-color gradient.
- A custom background and font color can be specified for when a row is selected.
- Custom zebra stripe colors can be added for alternate rows.
- Conditional background and font colors can be added per row based on table data.

5.6 Fewer Results in Lookup Controls

Lookup Combo Boxes, Lookup List Boxes and Lookup List Views are powerful controls to use in R:BASE forms for displaying records. However, filling the contents of the controls with too many records will decrease the speed in which the R:BASE form will load and refresh. It is also important to consider how many lookup controls are used on the form in relation to the number of records they will retrieve.

There are several ways to avoid populating Lookup controls with a great number of records:

1. Use the "Display Distinct Value(s)" check box within the object's settings.

2. Issue a DIALOG command before the form is loaded to prompt the end user for data specific to the results of the Lookup. Using the input within the Lookup's WHERE Clause will minimize the number of records returned.

3. Issue a DIALOG command before the form is loaded to prompt the end user for data specific to the results of the Lookup. Then, create a VIEW with resulting values and use the Lookup to display the resulting value(s).

5.7 Fewer Results in Pop-up Menus

Pop-up Menus offer the end user a pop-up dialog window to choose from displayed values. However, filling the contents of the pop-up with too many values will increase the amount of time the user must search for their desired value, and will exhaust time when the dialog loads the available options.

There are several ways to avoid populating the pop-up menu with too many values:

1. Use the "Distinct Value(s)" check box within the Pop-up Menu settings.

2. Issue a DIALOG command before the form is loaded to prompt the end user for data specific to the results of the pop-up menu. Using the input within the WHERE Clause settings of Pop-up Menu will minimize the number of records returned.

3. Add a dynamic WHERE Clause within the Pop-up Menu settings in order to prompt the user for a more specific result.

   Examples of dynamic WHERE Clauses:

   Example 01:

```
-- Dynamic WHERE Clause (Company LIKE)
Message:
Enter First Few Characters of Company
WHERE Clause:
WHERE Company LIKE '&%' ORDER BY Company
```

   Example 02:

```
-- Dynamic WHERE Clause (Company CONTAINS)
Message:
Enter Company Name:
WHERE Clause:
WHERE Company CONTAINS '&' ORDER BY Company
```

   Example 03:

```
-- Dynamic WHERE Clause (CustState =)
Message:
Enter State:
WHERE Clause:
WHERE CustState = '&' ORDER BY Company
```

Part VI

6 Operating in Single-User Mode

R:BASE is a multi-user system that can be used in single-user mode, which prevents other users from connecting to the database. Purpose of operating in single-user mode would be for a database administrator to perform structural changes or maintenance.

To operate in single-user mode:

1. Have all users disconnect from the database.
2. Enter SET MULTI OFF at the R> Prompt.
3. Reconnect one user to the database.

Increase Single-User Performance
If you do a process in a single-user environment, you will improve performance by using SET CLEAR OFF before executing commands that change or add rows to the database. SET CLEAR OFF sets up a 5K buffer to hold changes. Changes are written to disk when you SET CLEAR ON, when the buffer is full or is needed for the next page of data, or when you disconnect the database or exit from R:BASE. CLEAR can be set differently by individual workstations and is ignored in multi-user mode.

Part VII

7 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 VIII

8 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

- . -

- A -

- B -

- C -

- D -

- E -

- F -

- G -

- H -

- I -

- K -

- U -

- V -

- W -

Notes