R:BASE eXtreme 9.0

Managing Binary Large Objects

R:BASE Technologies, Inc.

Managing Binary Large Objects

by R:BASE Technologies, Inc.

Welcome to Managing Binary Large Objects!

One of the more exciting features in R:BASE eXtreme 9.0 for Windows is the ability to add, edit, delete, and display binary files-Binary Large OBjects, or BLOBs. With the advancements made to the R:BASE BLOB Editor, managing BLOB has never been easier!

Managing Binary Large Objects in R:BASE eXtreme 9.0

Printed: December 2009 in Murrysville, PA

First Edition

Table of Contents

Part I

1 Introduction

One of the more exciting features in R:BASE for Windows is the ability to add, edit, delete, and display binary files-**B**inary **L**arge **OB**jects, or BLOBs. This term refers to binary data as opposed to ASCII data. The data types VARBIT and BITNOTE were added for storing binary files within an R:BASE database. In addition, the VARCHAR data type lets you store large ASCII data files (**L**arge **OB**jects, or LOBs). The data for VARBIT and VARCHAR data types is stored in the fourth R:BASE data file, DBName.RB4 or DBName.RX4.

Part II

2 The Data Types

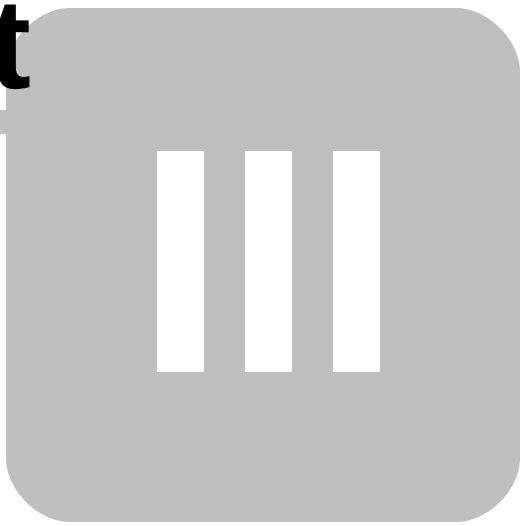
The VARBIT data type stores binary files-data such as bitmaps, charts, graphs, and logos. Any file can be stored in a VARBIT. The VARBIT data type is defined with a fixed length or as a variable length by using LONG VARBIT. A VARBIT can store a file up to 256MB in size per row. In general, it is easier to use the LONG VARBIT data type, which automatically deals with any size binary file. For very small binary files (less than 4,088 bytes), you can use the BITNOTE data type. A BITNOTE stores binary data in the DBName.RB2 or DBName.RX2 file similar to the way NOTE data is stored.

The VARCHAR data type stores large ASCII files-files with more data than can be placed in a NOTE data type. A rich text (RTF) document file from a word processing program such as Word or WordPerfect is a binary file, not an ASCII file. The document file contains formatting characters and must be saved as a text or ASCII file in order to be stored in R:BASE as a VARCHAR data type. As with the VARBIT data type, a VARCHAR data type is defined with a length, or as a LONG VARCHAR to be variable length.

Working with VARBIT and VARCHAR data types is different than working with other R:BASE data types. Normally, you think of loading data from one file into many columns and many rows in a table. With VARBIT and VARCHAR data, you are loading a file into one column in one row in a table. Once you have loaded a file into a VARBIT or VARCHAR data type, you can delete the original file from disk. The data file is now stored within the database.

Because you are loading a file into a column, it is often easiest to store VARBIT and VARCHAR data in a separate table linked to the data table with an ID column. The table storing the BLOB data can then include a column for the original file name and a column for a description of the BLOB data file.

Part



3 R:BASE BLOB Editor

In earlier Windows versions of R:BASE, the ability to add and manipulate large object data was relatively limited. However, with the latest Windows releases an integrated utility, the R:BASE BLOB Editor, has extended the functionality many times over. The R:BASE BLOB Editor is an integrated data utility to manage Large Objects (LOBs) and Binary Large OBjects (BLOBs) within R:BASE. With it, you can add, edit, or delete Binary Large Objects (BLOBs) within your database files.

The R:BASE BLOB Editor manages single and multi-page images, allowing users to manage multi-page images, such as .DCX, .GIF, or .TIFF files, when saved as BLOB data in R:BASE.

In addition to images, the R:BASE BLOB Editor also manages LOB data from the "Note/VarChar" and "Rich Text" tabs. This includes NOTE and VarChar data type columns within the database and Rich Text data inside and outside of the database.

Displaying the R:BASE BLOB Editor

When entering or editing BLOB/LOB data, the appropriate "Tab" of the R:BASE BLOB Editor is always automatically displayed when double-clicking on the NOTE, VARCHAR, BITNOTE, or VARBIT data type column objects.

1. When working with Data Browser/Editor and DB Grids;

Data Type	R:BASE BLOB Editor "Tab" Page
NOTE	Note/VarChar
VARCHAR	Image, Note/VarChar, Rich Text
BITNOTE	Image, Note/VarChar, Rich Text
VARBIT	Image, Note/VarChar, Rich Text

2. When working with Form Controls in Forms:

Form Control	R:BASE BLOB Editor "Tab" Page
DB Memo	Note/VarChar
DB Rich Edit	Rich Text
DB Image	Image

Loading BLOB/LOB Data from the Data Browser

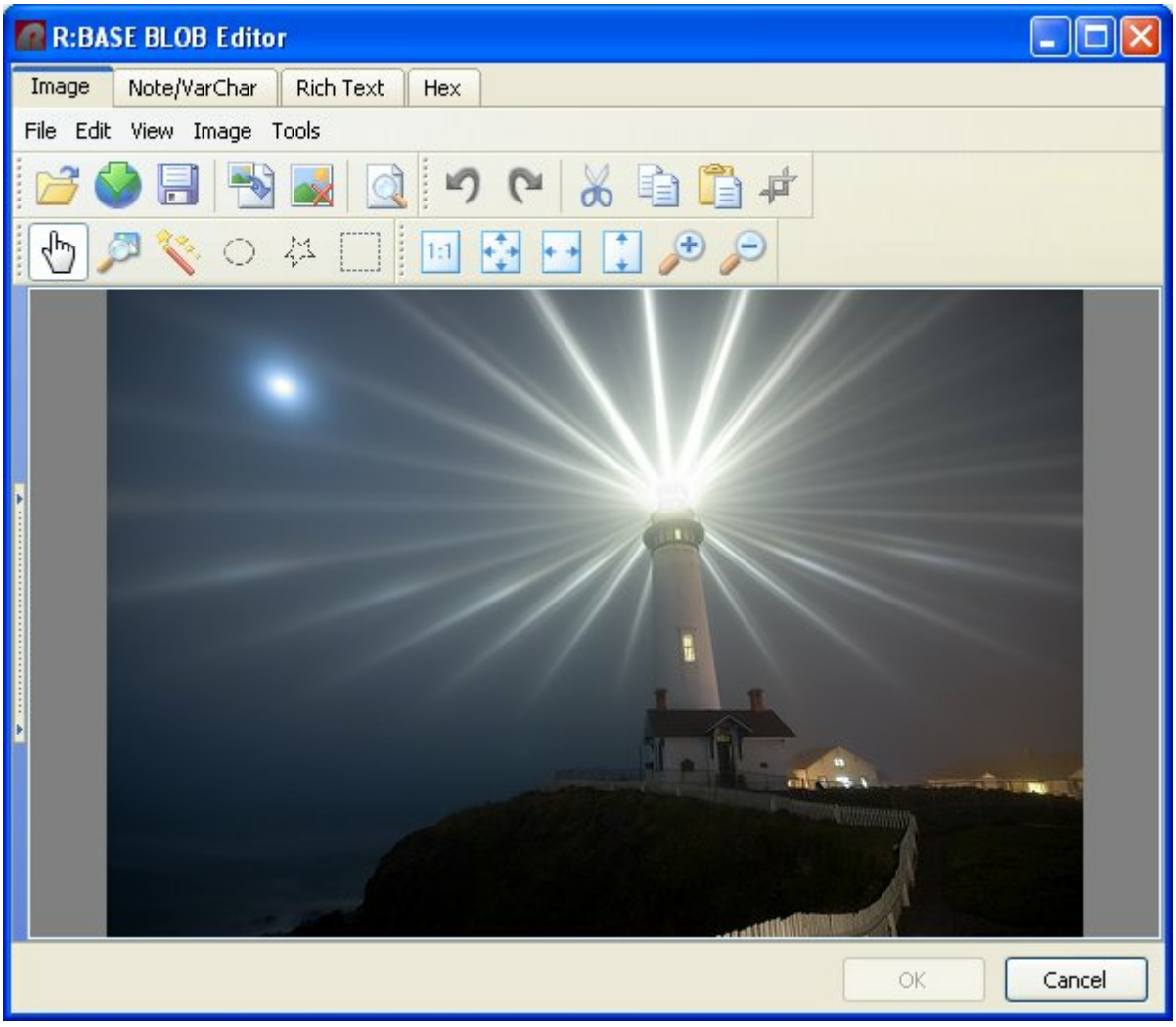
1. Open any table in the Data Browser where there are column(s) with VARBIT and/or VARCHAR data types.
2. Press the [F4] key to turn the DATA BROWSER into the DATA EDITOR.
3. Move your cursor focus to the column cell and double click on the field.
4. Select the appropriate tab for the data you want to load into the column.
5. Browse and select the appropriate image/data file and then click on the "Open" button.
6. The image/data file is now stored in the table row.
7. To verify, double click the field to open the "R:BASE BLOB Editor". Notice that the appropriate tab is specified based on the type of data file you loaded.

3.1 Image

You can add, edit, or delete images (BLOBs) within your database files. R:BASE eXtreme 9.0 recognizes the following file formats for images:

File Format	File Extension
TIFF Bitmap	tif, tiff, fax, g3n, g3f, xif
CompuServe Bitmap	gif
JPEG Bitmap	jpg, jpeg, jpe, jif
PaintBrush	pcx
Windows Bitmap	bmp, dib, rle

Windows Icon	ico
Windows Cursor	cur
Portable Network Graphic	png
Windows Metafile	wmf
Enhanced Windows Metafile	emf
Targa Bitmap	tga, targa, vda, icb, vst, pix
Potable Pixmap, GrayMap, BitMap	pxm, ppm, pgm, pbm
Wireless Bitmap	wbmp
JPEG2000	jp2
JPEG2000 Code Stream	j2k, jpc, j2c
Multipage PCX	dcx
Camera RAW	crw, cr2, nef, raw, pef, raf, x3f, bay, orf, srf, mrw, dcr, sr2
Photoshop PSD	psd
Video for Windows	avi
Mpeg	mpeg, mpg
Windows Media Video	wmv



Toolbar

Button	Description

	Open
	Load From URL
	File Save
	Insert Image from File
	Delete Selected Images
	Print Preview
	Undo
	Redo
	Cut
	Copy
	Paste
	Paste to Rectangle
	Scroll (mouse mode)
	Zoom (mouse mode)
	Magic Wand (mouse mode)
	Ellipse Selection (mouse mode)
	Polygon Selection (mouse mode)
	Rectangular Selection (mouse mode)
	Actual Size
	Fit Image
	Fit to Width
	Fit to Height
	Zoom In
	Zoom Out

Menu Bar

The Menu Bar within the "Images" tab provides many other advanced features to manage and manipulate images.

File

- *Open...* - opens a dialog window to browse and select an image file
- *Load from URL...* - loads an image from a URL
- *File Save* - saves the current image as an external file
- *Insert Image from File* - inserts image(s)
- *Delete Selected Images* - deletes the selected image(s)
- *Print Preview...* - opens the Print Preview dialog for changing print-specific settings
- *Select Source...* - opens a window to select or change TWAIN sources
- *Acquire Pages...* - starts the scan process of the current document

Edit

- *Undo* - undoes the last change made
- *Redo* - redoes the last "undo" operation

- *Cut* - cuts the current selection
- *Copy* - copies the current selection
- *Paste to Rectangle* - inserts or pastes the currently cut or copied selection

View

- *Histogram* - displays a graphical representation of the tonal distribution
- *Calculate Image Colors* - provides the total number of colors in the active image
- *Background Style* - displays a style for the image background
- *Background Color* - specifies the background color
- *Gradient End Color* - specifies the background gradient end color, when gradient is selected
- *Mouse Mode* - changes the mouse mode for image editing

Image

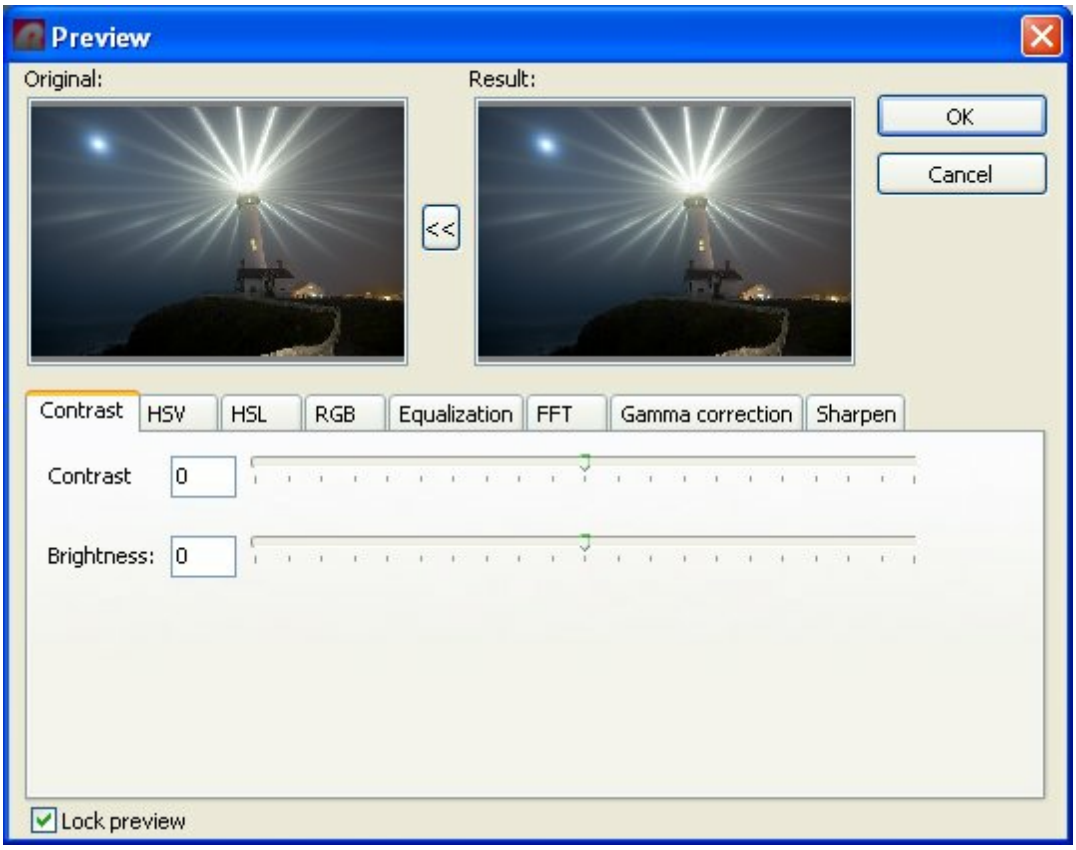
- *Color Adjust* - displays a dialog to adjust the color of the image
- *Negative* - replaces each pixel color with its opposite on the color wheel
- *Gray Scale* - converts the image to gray colors
- *Reduce Colors* - reduces the number of images used. A prompt for the new image number will appear.
- *Convert to BW* - converts the image to black and white colors
- *Convert to True Color* - converts the image to a 24-bit color image
- *Edge Detection* - marks the points at which the intensity changes sharply
- *Resize* - changes the size of the image
- *Resample* - change the resolution of an image up or down

Tools

- *Effects* - displays a dialog to add effects to the image
- *White Balance* - removes unrealistic color casts, so that objects in the image are rendered white just as in person
- *Rotate* - rotates an image around its center point
- *Vertical Flip* - flips an image vertically
- *Horizontal Flip* - flips an image horizontally
- *Remove Red Eyes* - removes "red eye" from photos
- *Equalize* - builds a histogram of colors used in all pixels in the image, from the brightest to the darkest, and then alters colors of pixels in between so there is the same number of pixels at all brightness levels
- *Blur* - reduces areas of high contrast and softens the appearance of an image
- *Zoom Properties* - applies a zoom filter

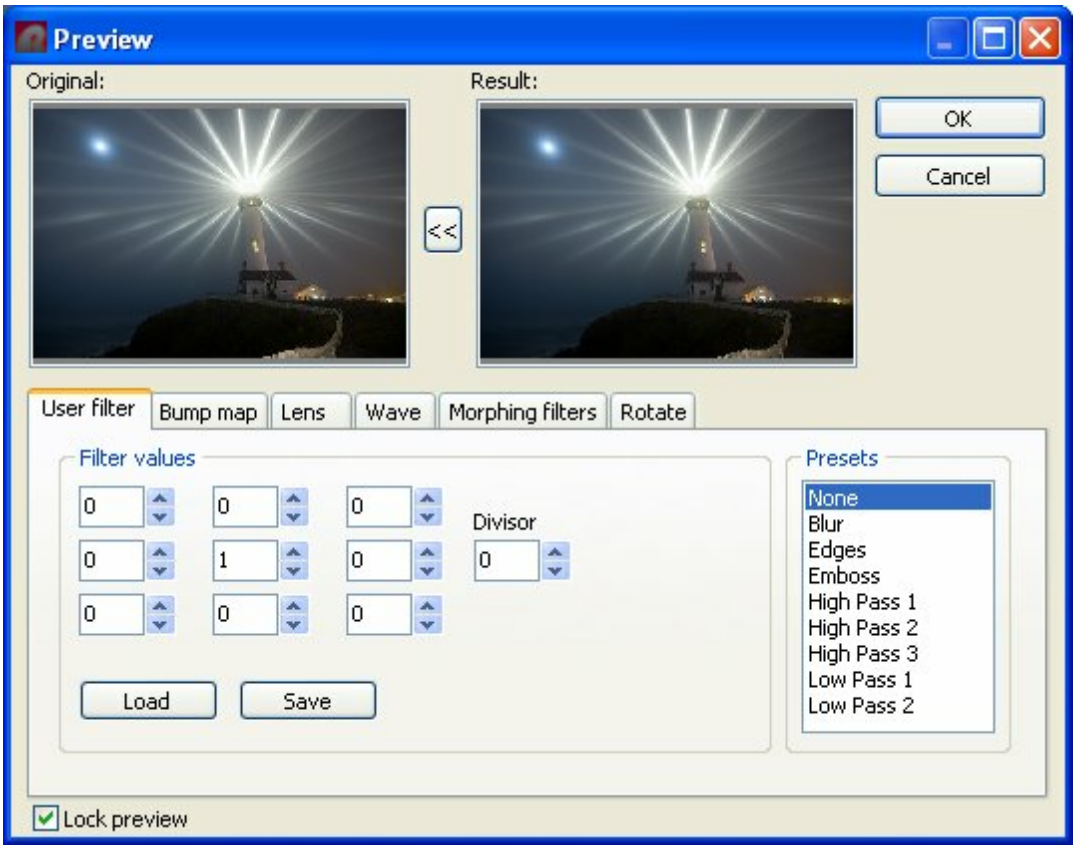
Color Adjust

When adjusting the color for an image, an original and resulting image is displayed to preview any changes before they are saved. To remove the automatic preview, unselect the "Lock preview" check box across the bottom of the window.



- *Contrast* - adjusts the Contrast and Brightness controls to lighten or darken the underlying layers and change the amount of shading, or contrast, between areas.
- *HSV* - adjusts the Hue, Saturation, and Value as related representations of points in an RGB color space
- *HSL* - adjusts the Hue, Saturation, and Luminosity as related representations of points in an RGB color space
- *RGB* - adjusts the Red, Green, Blue based upon the RGB color model
- *Equalization* - distributes the lightness values of the pixels more evenly across the light spectrum from black to white
- *FFT* - (Fast Fourier Transform) converts the image from the image (spatial) domain to the frequency domain
- *Gamma correction* - adjusts the the contrast and brightness in unison by using the red, green, and blue check boxes adjust the image color balance
- *Sharpen* - increases the contrast between adjacent pixels where there are significant color contrasts, usually at the edges of objects

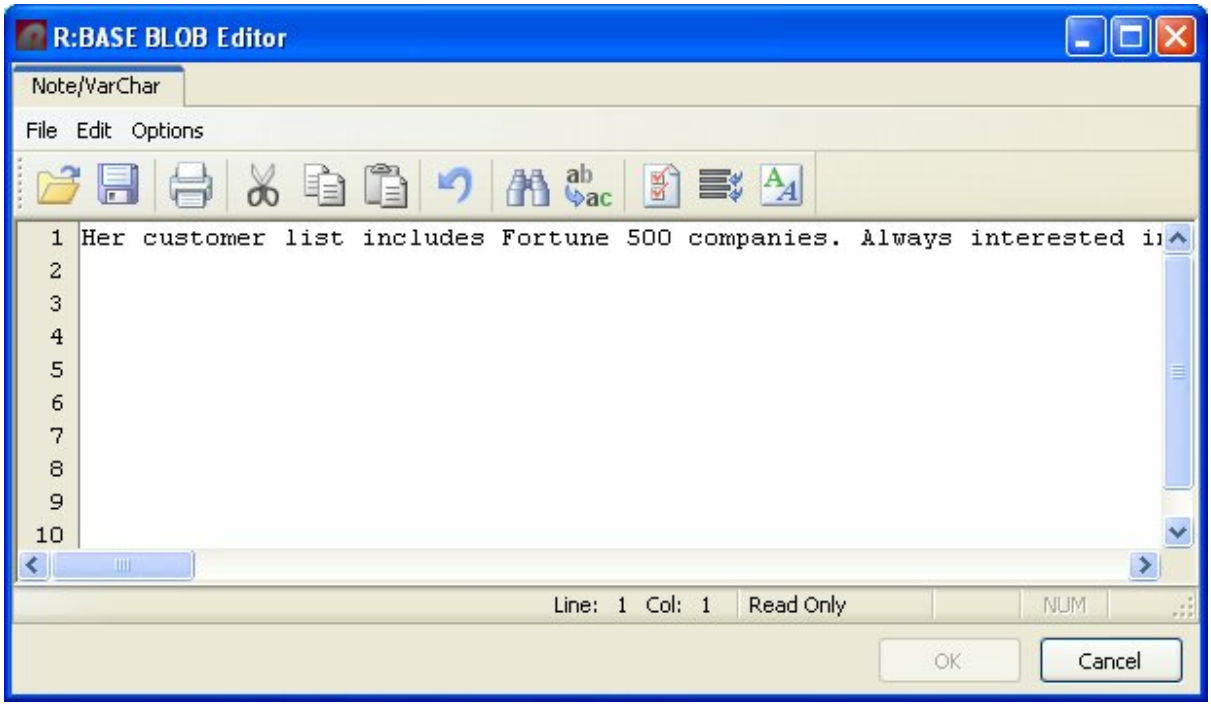
Effects



- *User filter* - provides filter presets, with the ability to load and save custom filters
- *Bump map* - creates a 3D effect by embossing an image and then mapping it to another image
- *Lens* - adds a lens filter to enhance depth
- *Wave* - distorts the image as if it had been disturbed by a number of waves
- *Morphing filters* - warps the image
- *Rotate* - rotates the image to an exact degree

3.2 Note/VarChar

You can load, edit, and save Note/VarChar data within R:BASE.

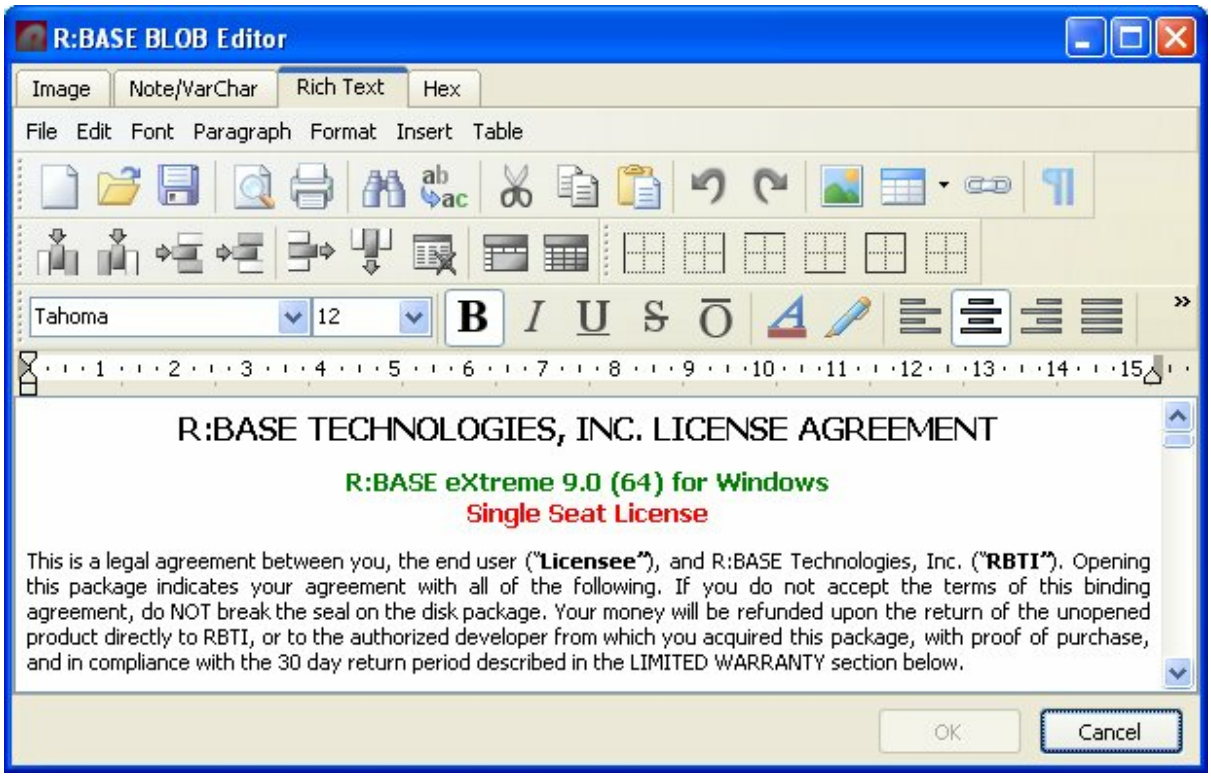


Toolbar

Button	Description
	Load From External File
	Save To External File
	Print
	Cut selected item
	Copy selected item
	Paste contents
	Undo the last action
	Search
	Search and Replace
	Show/Hide Right Margin
	Word Wrap
	Font...

3.3 Rich Text

You can load, edit, and save RTF data within R:BASE. The "RTF" tab of the R:BASE BLOB Editor allows many word processing capabilities for your RTF data.

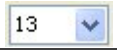
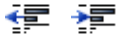


Toolbars

Button	Description
	New
	Open
	Save
	Print Preview
	Print
	Find
	Replace
	Cut
	Copy
	Paste
	Undo
	Redo

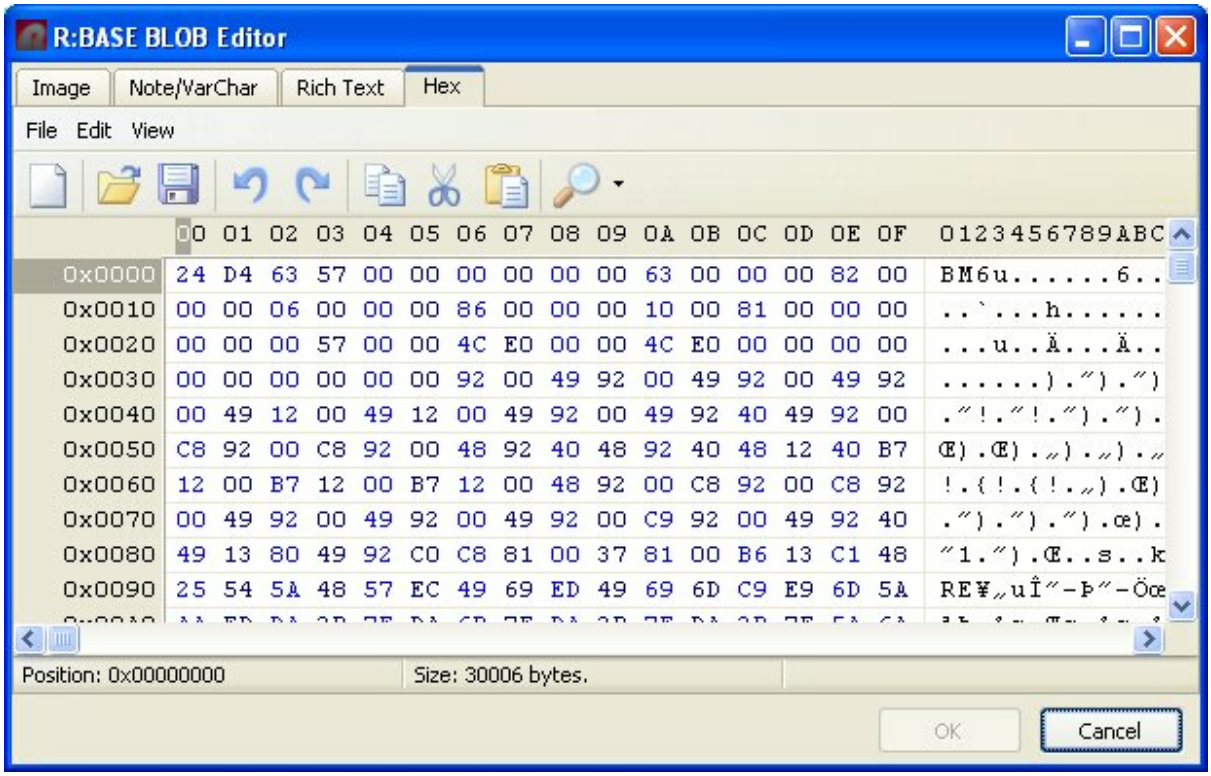
Button	Description
	Insert Column Left
	Insert Column Right
	Insert Row Above
	Insert Row Below
	Delete Column
	Delete Table
	Left Border
	Right Border
	Top Border
	Bottom Border
	All Borders
	No Borders

	Picture
	Insert Table
	Hypertext Link
	Nonprinting Characters

Button	Description
	Font Style
	Font Size
	**Bold**, *Italic,* Underline, ~~Strikeout~~, and Overline
	Left, Center, Right, and Full Justify
	Bullets
	Numbering
	Decrease Indent , Increase Indent
	Paragraph Background Color

3.4 Hex

The "Hex" tab allows the ability to see or edit the raw binary data.



Toolbar

Button	Description
	New
	Open
	Save
	Undo
	Redo
	Copy
	Cut
	Paste
	View

Part IV

4 Displaying BLOBs in Forms

Binary large objects can be displayed in forms by using the various form controls available in the Form Designer. The appropriate form control to use would depend on the type of binary large object you wish to display, and how the Binary large object is being made available to you.

Images

If you are displaying a static object, like a company logo, on a form, it is likely that the logo will remain there unchanged. In this instance, the Form "Image" control would be used to display the image.

If you are displaying a Form where images are stored for every row of the table, like employees records with photos, you can use the "DB Image" control. This allows the employee information and photo to display automatically when each row is viewed. When editing records from the Form, you can double click on the DB Image control to display the R:BASE BLOB Editor, where you can alter, delete, or load another image.

Another way to display image data on a Form is when there are far too many images to load into the database and you have the images located in a subdirectory of the database files. In this case, a "Variable Image" control can be used to hold the value of the path and image name. The most common way to link an image to a table record is to name the image file after the unique identifier column value. When using R:Charts, a Variable Image control is used to display the R:Chart image output.

Rich Text Format (RTF) Data

As with Images, the Form Designer supports controls for Rich Text Format data. For any Rich Text data stored in table records, you would use the "DB Rich Edit" control. For Rich Text data stored outside the database files or for displaying variables containing Rich Text data, you can use the "Variable Rich Edit" control.

Note/VarChar Data

Similar to Rich Text controls, the Form Designer supports controls for Note/VarChar data. For any Note/VarChar data stored in table records, you would use the "DB Memo" control. For Note/VarChar data stored outside the database files or for displaying variables containing Note/VarChar data, you can use the "Variable Memo" control.

Part V

5 Displaying BLOBs in Reports/Labels

Binary large objects can be displayed in Reports/Labels by using the various controls available in the Report/Label Designer. The appropriate Report/Label control to use would depend on the type of binary large object you wish to display, and how the Binary large object is being made available to you.

Images

If you are displaying a static object, like a company logo, on a report, it is likely that the logo will remain there unchanged. In this instance, the Report "Image" control would be used to display the image.

If you are displaying a Report where images are stored for every row of the table, like employees records with photos, you can use the "DB Image" control. This allows the employee information and photo to display on the printed document.

Another way to display image data on a Report is when there are far too many images to load into the database and you have the images located in a subdirectory of the database files. In this case, a "Variable Image" control can be used to hold the value of the path and image name. The most common way to link an image to a table record is to name the image file after the unique identifier column value. When using R:Charts, a Variable Image control is used to display the R:Chart image output.

Rich Text Format (RTF) Data

As with Images, the Report Designer supports controls for Rich Text Format data. For Rich Text data stored outside the database files, you can use the "Rich Text" or "Advanced Rich Text" controls. For any Rich Text data stored in table records, you would use the "DB Rich Text" or "Advanced DB Rich Text" controls. For displaying variables containing Rich Text data, you can use the "Variable Rich Text" or "Advanced Variable Rich Text" controls.

Note/VarChar Data

Similar to Rich Text controls, the Report Designer supports controls for Note/VarChar data. For Note/VarChar data stored outside the database files, you can use the "Memo" control. For any Note/VarChar data stored in table records, you would use the "DB Memo" control. For Note/VarChar data stored outside the database files or for displaying variables containing Note/VarChar data, you can use the "Variable Memo" control.

Part VI

6 Using Commands with BLOBs

The R:BASE command language is continuously being updated to support the many new features added to the product. With the support of BLOB and LOB data, several commands were directly enhanced.

6.1 BACKUP

Use the BACKUP command to copy the data and/or structure of a database to floppy disks or another directory on your hard disk.

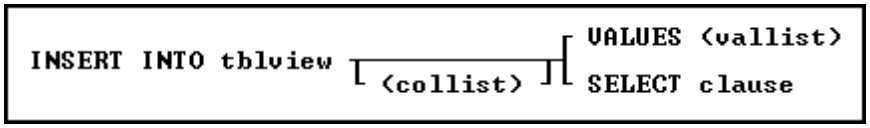
When backing up data for tables that contain binary large objects, the BACKUP command creates an additional file with a .LOB extension for the binary large objects. The originating file that you specify for the data and/or structure will share the same file name as the .LOB file. Your originating file can NOT have a .LOB file extension, otherwise, R:BASE will not be able to continue with the UNLOAD process.

The BACKUP command is very much like the UNLOAD command, only it supports the ability to span multiple diskettes when backing up databases to a floppy drive. With the many various backup software packages, or by simply performing file copies, the BACKUP command has become antiquated. It is recommended that users refer to the UNLOAD command for copying database structure and data.

For additional information and guidelines on the BACKUP command, please refer to the R:BASE Command Index within the main Help.

6.2 INSERT

The INSERT command is used to add data to a table or view without using a data-entry form. BLOB/LOB data can be loaded into a column directly from a file with the INSERT command. The file name is specified in a special format, **['filename.ext']**. This format tells R:BASE to find the specified file and treat it as BLOB data.



Options

(collist)
Specifies a list of one or more column names, separated by a comma (or the current delimiter). In an SQL command, any column name in the list can be preceded by a table or correlation name and a period (*tblname.colname*).

INTO tblview
Specifies the table or view name (views must be updatable).

SELECT clause
Finds values in a table, tables, or view to insert into the table or view specified by the INTO *tblview* option and the columns specified by the *collist* option.

VALUES (vallist)
Specifies a list of values to insert into the table specified by the INTO *tblview* option and the columns specified by the *collist* option. Separate values with a comma or the current delimiter.

For these data types...	**Use this format for *vallist***
All data types except BIT, BITNOTE, LONG VARBIT, and VARBIT	'string' or value
BIT, BITNOTE, LONG VARBIT, LONG VARCHAR, VARBIT, and VARCHAR	['filename.ext'] or ['filename.ext', filetype, offset, length] **Note:** When you use VARCHAR, the filetype is always TXT. When you use VARBIT, BIT, and BITNOTE, filetype

	refers to the standard graphical file types.

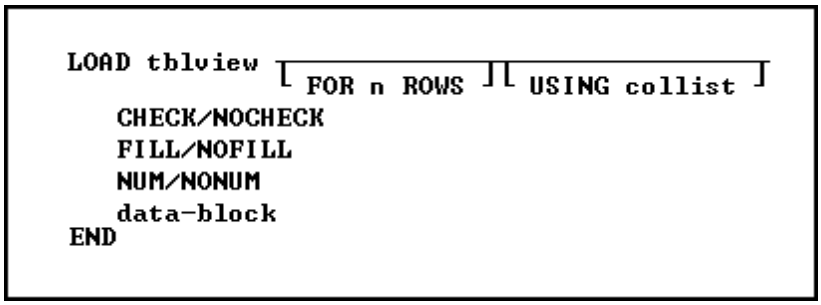
The following is an example of an INSERT command that adds a binary large object to a database:

```
INSERT INTO IMAGES (ID, IMAGEDATA) VALUES +
(1, ['filename.bmp'])
```

For additional information and guidelines on the INSERT command, please refer to the R:BASE Command Index within the main Help.

6.3 LOAD

The LOAD command is used to add data to a table or to a single table view that can be updated. BLOB/LOB data can be loaded into a column directly from a file with the LOAD command. The file name is specified in a special format, **['filename.ext']**. This format tells R:BASE to find the specified file and treat it as BLOB data.



Options

CHECK
NOCHECK
CHECK turns on rule checking. When rule checking is on, R:BASE checks input against data validation rules. NOCHECK turns off rule checking. CHECK and NOCHECK override the current setting of the SET RULES command. The default is CHECK.

data-block
Includes lines of data to be loaded, as well as the LOAD subcommands.

For these data types...	Use this format for *data-block*
All data types except BIT, BITNOTE, LONG VARBIT, and VARBIT	'string' or value
BIT, BITNOTE, LONG VARBIT, LONG VARCHAR, VARBIT, and VARCHAR	['filename.ext'] or ['filename.ext', filetype, offset, length] **Note:** When you use VARCHAR, the filetype is always TXT. When you use VARBIT, BIT, and BITNOTE, filetype refers to the standard graphical file types.

FILL
NOFILL
FILL makes null any columns that have not been assigned values. All of the missing values must be at the end of the row. If a rule specifies that a column requires an entry other than null, do not use FILL. NOFILL turns off FILL and requires a value for each column. The default is NOFILL.

FOR n ROWS
Directs R:BASE to stop processing after loading *n* rows, where *n* is a positive whole number. In the fourth syntax diagram, END is not used if FOR *n* ROWS is included.

FROM filespec
Loads data into the specified table with data from an external ASCII delimited file.

NUM
NONUM
NUM specifies that autonumbering columns will be numbered as they are loaded. NONUM turns off autonumbering while loading, thereby allowing loading of a specific value for autonumber columns. The default is NUM.

tblview
Specifies a table or view name to load.

USING collist
Specifies the column(s) to use with the command.

Loading a Data Block
The data block shown in the Syntax 4 diagram can include lines of data and any of the options available with LOAD-CHECK/NOCHECK, FILL/NOFILL, and NUM/NONUM. You can intersperse the options with data lines, and you can enter more than one option on a line if you separate the options with semicolons. However, you cannot combine data and options on the same line.

R:BASE displays the dialog prompt to accept data-block entry. LOAD adds data to a table, row by row, without using a data-entry form and without prompting for each data item.

You can enter the options for the LOAD command at the dialog prompt at any time during data loading. Or you can include them on the command line, separated from the command by semicolons, as shown in the example below. (Do not use this format in command or procedure files. All options must follow the LOAD command on separate lines in command or procedure files.)

```
LOAD transdetail ; CHECK ; NUM
```

You can use global or system variables instead of constant values in the data block.

To enter values properly, use the following guidelines.

- Enter column values in the order that columns are defined in the table, and separate the values with a delimiter character. The default delimiter character is the comma.
- You can enter up to 75 characters on a single line. If a row is longer than 75 characters, continue on to the next line by typing past the end of the current line or by entering a plus character at any point. The plus character must be the last entry on the line. If you are using this form of the LOAD command in a command file, you must use a plus character to continue on the next line; the lines will not automatically wrap.

Example:

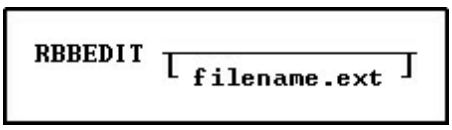
The following is an example of an LOAD command file that adds a binary large object to a database:

```
LOAD Images USING ImageData
  NUM
  ['filename.bmp']
END
```

For additional information and guidelines on the LOAD command, please refer to the R:BASE Command Index within the main Help.

6.4 RBBEDIT

The RBBEDIT command is used to launch the R:BASE BLOB Editor, with which you can create or edit binary files.



You can use the RBBEDIT command to launch the R:BASE BLOB Editor for managing external files at the R> Prompt, in a command file, or in an EEP. Depending on the file extension, the R:BASE BLOB Editor will display the available options accordingly.

Options

filename.ext
Specifies an external ASCII text, Rich Text, or image file. The R:BASE BLOB Editor recognizes the following image formats:

File Format	File Extension
TIFF Bitmap	tif, tiff, fax, g3n, g3f, xif
CompuServe Bitmap	gif
JPEG Bitmap	jpg, jpeg, jpe, jif
PaintBrush	pcx
Windows Bitmap	bmp, dib, rle
Windows Icon	ico
Windows Cursor	cur
Portable Network Graphic	png
Windows Metafile	wmf
Enhanced Windows Metafile	emf
Targa Bitmap	tga, targa, vda, icb, vst, pix
Potable Pixmap, GrayMap, BitMap	pxm, ppm, pgm, pbm
Wireless Bitmap	wbmp
JPEG2000	jp2
JPEG2000 Code Stream	j2k, jpc, j2c
Multipage PCX	dcx
Camera RAW	crw, cr2, nef, raw, pef, raf, x3f, bay, orf, srf, mrw, dcr, sr2
Photoshop PSD	psd
Video for Windows	avi
Mpeg	mpeg, mpg
Windows Media Video	wmv

Examples:

01.
The following will display the "Image" tab of the R:BASE BLOB Editor for you to load an image.

```
RBBEDIT
or
RBBE
```

02.
The following will display a bmp image in the "Image" tab of the R:BASE BLOB Editor.

```
RBBEDIT myphoto.bmp
```

03.
The following will display the text file in the "Note/VarChar" tab of the R:BASE BLOB Editor.

```
RBBEDIT mytext.txt
```

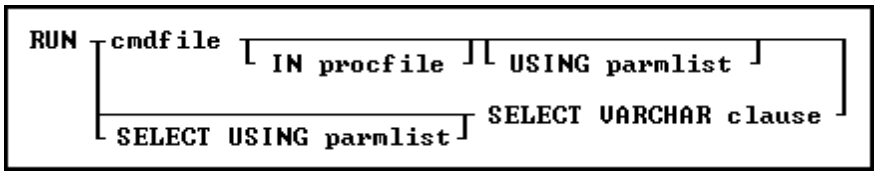
04.
The following will display the "Rich Text" tab of the R:BASE BLOB Editor for you to create a new RTF file.

```
RBBEDIT myfile.RTF
```

This command will also create a new file if one is not already defined.

6.5 RUN

Use the RUN command to run command blocks, command files, and command files requiring passed parameters.



Options

cmdfile
Specifies the name of the command block or command file to execute.

On a workstation with multiple drives (local or mapped), especially when the files are on the different drive, it is always the best practice to define a drive letter when copying, deleting, renaming or running files, unless the specified files are located in the working directory. You will not need to specify the drive letter if all of the files are located in the default directory when using the copy, delete, rename or run commands.

IN procfile
Specifies the name of a procedure file. A procedure file is a compiled binary file that contains stored menu, screen, and command blocks. Include a drive and path name if the procedure file is not on the current drive and directory.

SELECT VARCHAR clause
Specifies a column defined with the VARCHAR data type from a table, from which you can run the contents. The SELECT clause must limit the data to only one row; otherwise, an error is returned.

USING parmlist
Lists the values the command file uses when it runs. The parameter list can contain up to 18 values. The first value in the list is referenced in the executed file as %1, the second as %2, and so on through %9. They are treated just like other variables. To reference the contents of these variables, preface the variable name with a dot (.); for example, set *v1* =.%1.

About the RUN Command

The RUN command must be on a line by itself and not combined with other commands.

Examples

The following command runs a file named MYCMD.CMD in the current working directory.

```
RUN mycmd.cmd
```

The following command runs a command block named *mycmdfil*in the MYPROCFL.APX procedure file.

```
RUN mycmdfil IN myprocfl.apx
```

The following command executes the *mycmdfil* command block in the MYPROCFL.APX procedure file, placing the parameter values, *Display This Message* and *10*, into parameter variables %1 and %2, respectively.

```
RUN mycmdfil IN myprocfl.apx USING 'Display This Message' 10
```

The following commands, which use the system variable *#time*, make up a timing procedure called *mycmdfil.* This procedure displays the message passed as parameter *%1* for the length of time indicated in parameter *%2*. If *mycmdfil* is added in a procedure file named MYPROCFL.APX, the RUN command causes *mycmdfil* to display the message for 10 seconds.

```
$COMMAND
mycmdfil
SET VARIABLE vstart TIME = .#TIME, vwait INTEGER = 0
WHILE vwait < .%2 THEN
   SHOW VARIABLE %1 AT 10 20
   SET VARIABLE vwait = (.#TIME - .vstart)
ENDWHILE
CLS FROM 10 TO 10
CLEAR VARIABLES vstart, vwait, %1, %2
RETURN
```

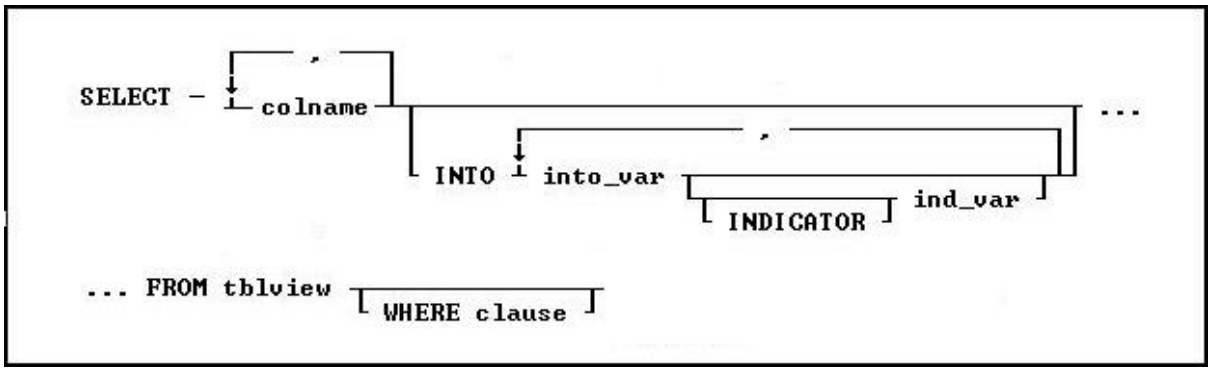
The following RUN command executes the command syntax with the *CmdData* VARCHAR data field inside the table *IntrnlCmd.*

```
RUN SELECT CmdData FROM IntrnlCmd WHERE CmdName = 'NewKey'
```

For additional information and guidelines on the RUN command, please refer to the R:BASE Command Index within the main Help.

6.6 SELECT

Use the SELECT command to display rows of data from a table or view. To display the data in the order you want, modify the SELECT command by using various clauses.



Options

,
Indicates that this part of the command is repeatable.

colname
Specifies a column name. In R:BASE eXtreme 9.0 (32), the column name is limited to 18 characters. In R:BASE eXtreme 9.0 (64), the column name is limited to 128 characters.

FROM
Lists the tables from which data is to be displayed.

ind_var
Specifies a variable result indicator to be used with an INTO clause in a SELECT command. This variable stores the status of the variable: non-null (0) or null (-1).

INDICATOR
Indicates the following variable is an indicator variable, which is used to indicate if a null value is retrieved.

INTO
Selects information directly from a table and puts it into variables. You must include a WHERE clause so the SELECT command finds only one row.

into_var
Specifies a variable whose value is assigned with an INTO clause in a SELECT command.

tblview
Specifies a table or view name.

WHERE clause
Limits rows of data. For additional information and guidelines on the WHERE clause, please refer to the R:BASE Command Index within the main Help.

About the SELECT INTO command

This optional clause loads the results of a SELECT command into variables, but does not display the results on screen.

An INTO clause loads the resulting value of each column, expression, or function included in the command clause into a variable. If previous clauses have returned more than one row, the values assigned to the variables are unpredictable. You should make sure you are returning only one row. Either test the results before using an INTO clause or check the value of the variable *sqlcode* after executing the command. If the clause is successful, *sqlcode* is 0.

Comments

The INTO clause must have a corresponding variable for every item in the command clause; values are assigned to variables in the order of items in the command clause. The data type of each command clause item and its corresponding *into_var* must be compatible. For example:

```
SELECT MAX(listprice), MIN(listprice) +
   INTO vmaxprice INDICATOR vind_max, +
              vminprice INDICATOR vind_min +
                   FROM product
```

The MAX and MIN functions assign the value $3,100.00 to the variable *vmaxprice* and $1,900.00 to *vminprice*. These values are the maximum and minimum values for the *listprice* column in the *product* table. Since both functions returned values, the value of both indicator variables is 0.

Example

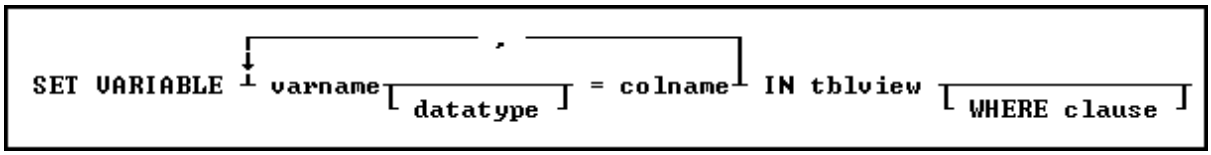
The following example sets the variable *vImage* to the value of the *ContPhoto* image in the *Contact* table:

```
SELECT ContactPhoto INTO vImage FROM Contact WHERE ContID = 1007
```

For additional information and guidelines on the SELECT command, please refer to the R:BASE Command Index within the main Help.

6.7 SET VARIABLE

Use the SET VARIABLE command to define or redefine a variable value and/or data type. In an application program, you can use the SET VARIABLE command to define a variable that is equal to the file name that contains the binary large object.



Options

,
Indicates that this part of the command is repeatable.

colname
Specifies a column name. In R:BASE eXtreme 9.0 (32), the column name is limited to 18 characters. In R:BASE eXtreme 9.0 (64), the column name is limited to 128 characters.

datatype
Specifies an R:BASE data type for the variable.

IN tblview
Specifies a table or view.

varname
Specifies a variable name. In R:BASE eXtreme 9.0 (32), the column name is limited to 18 characters. In R:BASE eXtreme 9.0 (64), the column name is limited to 128 characters.

WHERE clause
Limits rows of data. For additional information and guidelines on the WHERE clause, please refer to the R:BASE Command Index with the main Help.

Assigning a Data Type to a Variable
The *datatype* option refers to one of the valid R:BASE data types: BIT, BITNOTE, CURRENCY, DATE, DATETIME, DOUBLE, INTEGER, LONG VARBIT, LONG VARCHAR, NOTE, NUMERIC, REAL, TIME, VARBIT, OR VARCHAR. You can define a variable to have a NOTE data type, but R:BASE treats it as TEXT for most uses. You can also specify the precision and scale for NUMERIC data types.

The *datatype* option creates a variable, determines its data type, and sets its value to null. Use this option to define a variable's data type before assigning a value to the variable. To set multiple variables in the same command, separate the variables by a comma or the current delimiter.

Assigning Column Values in a Table or View
If you specify a table or view in a SET VARIABLE command, you can include an optional WHERE clause to indicate which row to use. If you do not include the WHERE clause, R:BASE uses the value for the column in the first row.

You must have SELECT privileges on the table to use this form of SET VARIABLE.

You can define multiple variables with a single SET VARIABLE command when you set the value of the variables to the value of columns in a table. However, when capturing column data into variables, it is better to use the SELECT command; specifically, SELECT INTO. SELECT INTO is the SQL compliant command when capturing table data into variables.

Example

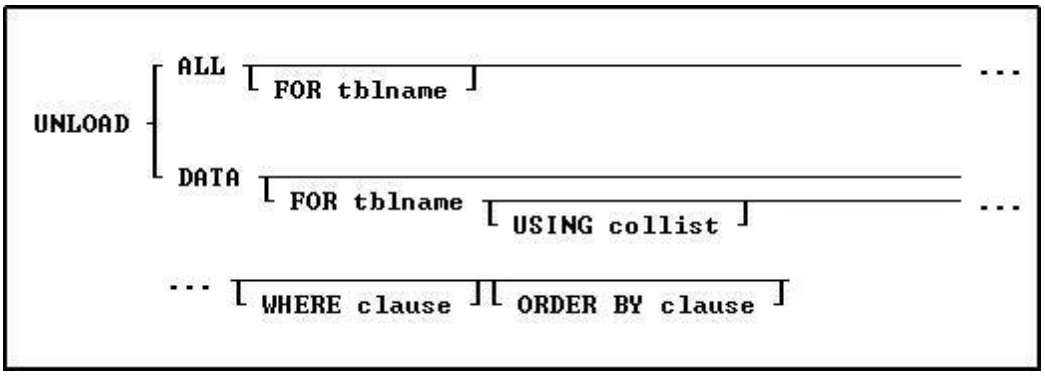
The following example sets the variable *vImage* to the value of the *ContPhoto* image in the *Contact* table:

```
SET VAR vImage VARBIT = ContactPhoto IN Contact WHERE ContID = 1007
```

For additional information and guidelines on the SET VARIABLE command, please refer to the R:BASE Command Index within the main Help.

6.8 UNLOAD

Use the UNLOAD command to copy the data, structure, or data and structure of a database or table to a specified output device. UNLOAD can be used to transfer tables or views from one database to another, or to back up a database.



Options

ALL
Unloads both the data and the structure.

DATA
Unloads just the data.

FOR tblview
Specifies a single table/view to unload the SQL command necessary to create a specific table/view.

ORDER BY clause
Sorts rows of data. For additional information and guidelines on the ORDER BY clause, please refer to the R:BASE Command Index within the main Help.

tblname
Specifies the table name to unload the data, structure, or both.

USING collist
Specifies the column(s) to use with the command.

WHERE clause
Limits rows of data. For additional information and guidelines on the WHERE clause, please refer to the R:BASE Command Index within the main Help.

About the UNLOAD Command

The UNLOAD ALL and UNLOAD STRUCTURE commands require the database owner's user identifier if the database has had access rights granted with the GRANT command. R:BASE places the owner's user identifier and all the granted access rights in the file created by UNLOAD to ensure that the restored database file continues to be protected. Be sure to protect this file from unauthorized users.

When unloading data for tables that contain binary large objects, the UNLOAD command creates an additional file with a .LOB extension for the binary large objects. The originating file that you specify for the data and/or structure will share the same file name as the .LOB file. Your originating file can NOT have a .LOB file extension, otherwise, R:BASE will not be able to continue with the UNLOAD process.

Example

```
R> OUTPUT Contact.ALL
R> UNLOAD ALL FROM Contact
R> OUTPUT SCR
R> DIR Contact.*
```

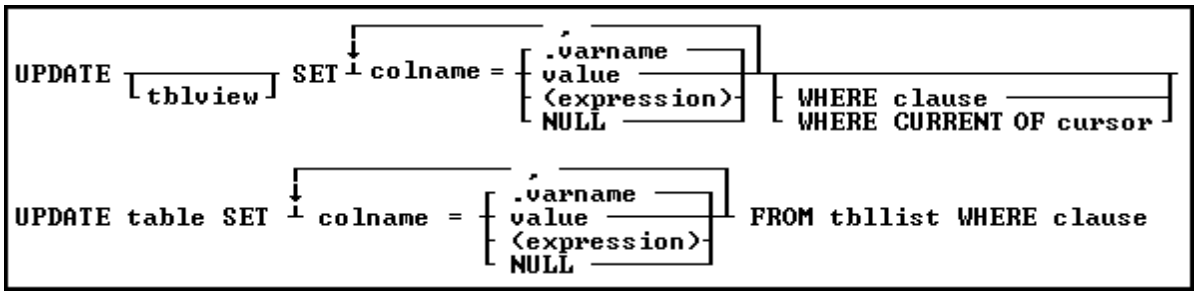
```
 Directory of C:\RBTI\RBG76\Samples\RRBYW14\

Contact  ALL       10442   7-10-08   3:53p Contact.ALL
Contact  LOB     1407089   7-10-08   3:53p Contact.LOB
        2 File(s)          1417531 bytes
                       53813653504 bytes free
```

For additional information and guidelines on the UNLOAD command, please refer to the R:BASE Command Index within the main Help.

6.9 UPDATE

Use the UPDATE command to change the data in one or more columns in a table or a view. BLOB/LOB data can be added or replaced from a file or variable using the UPDATE command.



Options

,
Indicates that this part of the command is repeatable.

(expression)
Determines a value using a text or arithmetic formula. The expression can include other columns from the table, constant values, functions, or system variables such as *#date*, *#time*, and *#pi*.

FROM tbllist
Specifies a list of tables from which data can be retrieved and updated.

NULL
Sets the values in the column equal to null.

SET colname
Specifies the column to update.

table
Specifies a table.

tblview
Specifies a table or view. If no table or view name is included, columns will be updated in all tables containing the specified columns, according to the conditions of the WHERE clause.

value
Specifies a value to enter in the specified column.

.varname
Specifies a global variable that provides a value for a column.

WHERE clause
Limits rows of data. For more information, see the "WHERE" entry.

WHERE CURRENT OF cursor
Specifies a cursor that refers to a specific row to be affected by the UPDATE command. With this option, you must specify *tblview*.

About the UPDATE Command

This command is useful for adjusting values in columns that require uniform changes.

The UPDATE command only modifies data in columns in one table or view. You can also update a table by referencing values from another table. You can modify a column's value by doing the following:

- Entering a new value for the column as a constant or variable
- Entering an expression that calculates a value for the column
- Entering a null value

Notes:

- Only users that have been granted rights to update the table(s) or column(s) can run the UPDATE command.
- R:BASE complies with defined rules, even for columns not affected by the update. If an update breaks a rule, the update is not processed.
- You cannot use UPDATE with computed or autonumbered columns. To change a computed column value, change the values in the columns to which the computed column refers.
- The UPDATE command will not update data in a multi-table View (a View based on multiple tables), as the data is not editable.
- A View with a GROUP BY parameter is also not editable.

Updating Column Values

You can update a column with a specific value. The value you use must meet the requirements of the column's data type, for example, a numeric column cannot be loaded with a text value.

Use the current delimiter character (the default is a comma) to separate each column and its new value from the next column and value.

Use the following guidelines when modifying data with UPDATE:

- Do not embed commas within entries for CURRENCY, DATE, DATETIME, DOUBLE, INTEGER, NUMERIC, or REAL data types. R:BASE automatically inserts commas and the current currency symbol.
- When values for CURRENCY, DOUBLE, NUMERIC, or REAL or data types are decimal fractions, you must enter the decimal point. When values are whole numbers, R:BASE adds a decimal point for you at the end of the number. R:BASE adds zeros for subunits in whole currency values. For example, using the default currency format, R:BASE loads an entry of 1000 as $1,000.00.
- When values for NOTE or TEXT data types contain commas, you can either enclose the entries within quotes, or use SET DELIMIT to change the default delimiter (comma) to another character.
- When values for NOTE or TEXT data types contain single quotes ('), and you are using the default QUOTES character ('), use two single quotes ('') in the text string. For example, 'Walter Finnegan''s order.'
- When values for NOTE or TEXT data types exceed the maximum length of a column, R:BASE truncates the value and adds it to the table. A message is displayed that tells you which row has been truncated.

Using an Expression or Variable

Enclose expressions in parentheses. If you use global variables in an expression, dot the variable (. *varname*). If expressions contain values that have a TEXT data type, enclose the values within quotes. The default QUOTES character is the single quote (').

If you attempt to use a null value in an expression or computed column, the result of the expression is null. However, if you set ZERO to on, R:BASE treats null values as zeros and processes expressions as if the null value were zero.

Using the WHERE Clause

If an UPDATE command includes a table or view name, you do not need to specify a WHERE or WHERE CURRENT OF clause. All rows will be updated.

If you use a WHERE CURRENT OF clause, you must include a table or view name in the command.

If you omit a table or view name, you must use a WHERE clause with the UPDATE command so that you do not change values in more rows than you intended to change. The WHERE clause pinpoints the rows you want to change. If any columns exist in more than one table, all occurrences are changed if the column value meets the WHERE clause conditions. Test the WHERE clause by using the SELECT command before using the clause with UPDATE command. By using a WHERE clause with a SELECT command, you can view the rows you want to change before changing them.

R:BASE takes significantly less time to process a WHERE clause if one of the columns specified in the clause is an indexed column.

Examples

The following example updates a VARBIT column value with a variable *vImage.*

```
R> SET VAR vImage VARBIT = ContactPhoto IN Contact WHERE ContID = 1007
R> UPDATE Contact SET ContactPhoto = .vImage WHERE ContID = 1009
```

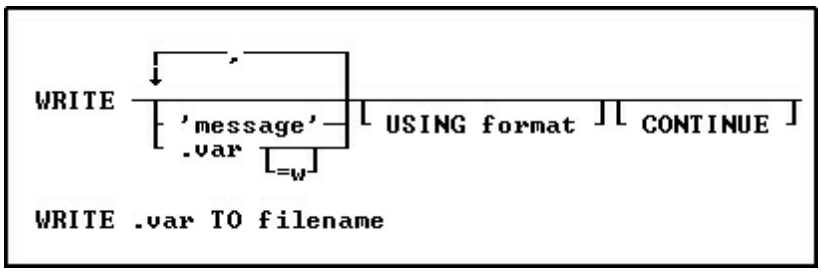
The following example updates a VARBIT column value with image file.

```
R> UPDATE Contact SET ContPhoto = ['JimMiller.bmp']  WHERE ContID = 1010
```

For additional information and guidelines on the UPDATE command, please refer to the R:BASE Command Index within the main Help.

6.10 WRITE

Use the WRITE command to send a message to an output device or to display a message on R> Prompt screen. In this context, the WRITE command is primarily used with R:BASE for DOS. To display messages in R:BASE for Windows, use the PAUSE command. Another use for the WRITE command is to recreate binary or large ASCII data to a file.



Options

,
Indicates that this part of the command is repeatable.

'message'
Specifies a message to be displayed on the screen.

filename.ext
The name of the file and extension for writing variable BLOB data to a file.

USING format
Specifies a display format for the message or variable.

.var
Specifies a dot variable of any data type, which can be used instead of a message.

=w
Specifies the display width for the variable.

About the WRITE Command

After you place binary or large ASCII data into your R:BASE database, the The original file can be recreated at any time using the WRITE command. The data is read from the table into a variable, then the variable is written to a file. This process recreates the file exactly.

Example

```
SET VAR vImage VARBIT = ContactPhoto IN Contact WHERE ContID = 1007
WRITE .vImage TO Contact1007.jpg
```

For additional information and guidelines on the WRITE command, please refer to the R:BASE Command Index within the main Help.

Part VII

7 Sample Use of BLOBs

Store Application Files and Command Files in a Database

Using the VARCHAR and VARBIT data types, you can store command and application files within the database itself, making it easier to distribute and backup applications. In addition, a description of each command file can be stored with the file for easy and convenient documentation of applications. Application files (.RBA), CODELOCKed procedure files (.APX), ASCII command files, and EEPs can be stored.

Even though the VARBIT data type can accommodates both ASCII and binary (.RBA and CODELOCKed) files, the ASCII files will be stored in a VARCHAR data type column, while the Application files and CODELOCKed procedure files will be stored in a VARBIT data type column in order to retain a ASCII version of CODELOCKed files. Add a table to the database to store the application files.

```
CREATE TABLE AppTable (AppID INTEGER, +
AppName TEXT 20, RBFileName TEXT 30, +
FileDescription NOTE, ModifiedDate = (.#DATE) DATE, +
RawFileData VARCHAR, BinFileData VARBIT)
AUTONUM AppID IN AppTable USING 1,1
```

Column	Description
AppID	An INTEGER autonumber column provides a unique identifier for each row.
AppName	The application name. Different applications can be stored in the same table.
RBFileName	The R:BASE application or command file name for the data loaded into the RBFileData column on this row. The name is used to recreate the application or command file for execution.
FileDescription	A description of the application or command file.
ModifiedDate	A computed date column that is automatically updated whenever data in the row changes. This lets you know if the most recent copies of the command files are stored.
RawFileData	The raw ASCII command file data
BinFileData	The binary command file data

You can load the table using INSERT commands from the R> Prompt, by launching the Data Browser, or by running a short command file. Using a command file is more efficient for applications made up of many files. The command file quickly loads all the files and file names. The sample command file below, loadfile.rmd, prompts for a file extension, and automatically loads all selected files. It loads data into the RBFileName and RawFileData columns only. The AppID and ModifiedDate columns are automatically loaded. The other columns are edited after the files are loaded. The command file uses a CHOOSE command to select the files and return a comma-delimited list. A WHILE loop retrieves the files one by one, formats the name for loading into a VARCHAR data type, and loads the file.

```
--loadfile.rmd
CLS
SET MESSAGE OFF
SET ERROR MESSAGES OFF
SET ERROR VAR vError
SET VAR vExt TEXT = NULL, vAllNames TEXT = NULL

-- prompt for the file extension to load. For example,
--.RMD, .EEP, or .APP. Wild cards can be used.
DIALOG 'Enter file extension (ASCII only):' vExt=3 vKey 1 +
CAPTION 'Wildcards Accepted!' +
ICON INFO OPTION MESSAGE_FONT_COLOR 536870911 +
|MESSAGE_BACK_COLOR -16777201 +
```

```
|MESSAGE_FONT_NAME MS Sans Serif +
|MESSAGE_FONT_SIZE 10|MESSAGE_BOLD ON

-- create a file reference to use in the CHOOSE command
SET VAR vExt1 = ('*.'+ .vExt)

-- bring up a menu of all files in the current directory
-- matching the specified extension.
CHOOSE vAllNames FROM #LFILES IN &vExt1 CHKBOX +
TITLE 'File Selection' +
CAPTION 'Press [Shift]+[F6] to Select All files...' +
LINES 15 OPTION BUTTONS_SHOW_GLYPH ON +
|TITLE_BOLD ON|TITLE_FONT_SIZE 10

-- exit if nothing selected or no files found
IF vError <> 0 OR vAllNames = '[Esc]' THEN
  RETURN
ENDIF

-- get the first file name
SET VAR vCount INTEGER = 1
SET VAR vFName1 = (SSUB(.vAllNames, .vCount))

-- format the file name for loading into a VARCHAR column
-- the file name needs surrounded by brackets and quotes
SET VAR vFName2 = ('[''' + .vFName1 + ''']')
PAUSE 3 USING 'Processing files'
WHILE vFName1 IS NOT NULL THEN

  -- load the file name and the file data, you must reference
  -- the formatted file name as an ampersand variable
  INSERT INTO AppTable (RBFileName, RawFileData) VALUES +
  (.vFName1, &vFName2)

  -- get the next file name
  SET VAR vCount = (.vCount + 1)
  SET VAR vFName1 = (SSUB(.vAllNames, .vCount))
  SET VAR vFName2 = ('[''' + .vFName1 + ''']')
ENDWHILE
CLS
RETURN
```

After all the command files and application files have been loaded, open the table, AppTable, with the Data Browser and fill in the data for the remaining columns, AppName and FileDescription. Make sure you have switch the "Edit Mode" in the Browser. For any binary application files, double click on the BinFileData column to launch the R:BASE BLOB Editor and load your binary files. Within the Data Browser, you can also double click on the RawFileData column to edit the contents of your command files. This functionality avoids you having to save the file to disk, edit the contents, and then load the file back into the table.

The command files can be enhanced to prompt for an application name, or to prompt for a specific file. The process is the same; put the file data into a variable, and then write the data to a file. Once the table has been loaded with all of the command files and application files, the database is ready for testing and distribution, with only a small command file needed with the database files.

At any time when you are using the database, the ASCII command files can be RUN from their stored location, without you having to create the file to disk. The RUN SELECT command allows this. The following example will run the ASCII command file stored in the first row of the AppTable:

```
RUN SELECT RawFileData FROM AppTable WHERE AppID = 1
```

If you need to recreate the command files or application files as disk files from their stored location in the database, then, at any time, the files can be easily restored.

```
--bldfile.rmd
-- a cursor on AppTable walks through each row, making
-- a disk file for each file that has been loaded

SET VAR vChkCurs1 INTEGER = (CHKCUR('c1'))
IF vChkCurs1 = 1 THEN
  DROP CURSOR c1
ENDIF

DECLARE c1 CURSOR FOR SELECT RBFileName, RawFileData FROM AppTable
OPEN c1

-- place the file name and the file data into variables
FETCH c1 INTO vFileName INDICATOR iv1, vFileData INDICATOR iv2
WHILE SQLCODE <> 100 THEN

  -- place the data back into a disk file.
  -- The file is recreated exactly
  WRITE .vFileData TO .vFileName

  -- get the next file
  FETCH c1 INTO vFileName INDICATOR iv1, vFileData INDICATOR iv2
ENDWHILE
CLOSE c1
DROP CURSOR c1
CLS
RETURN
```

If a replacement command file or application file needs updated to the AppTable, use a command file to update the files with files from the disk.

```
--updfile.rmd
-- prompt for the file type
SET VAR vFileName = NULL

DIALOG 'Is this an ASCII or Binary file?' vFileType vEndKey YES +
CAPTION 'Choose data type' ICON QUESTION +
OPTION MESSAGE_FONT_COLOR 536870911 +
|MESSAGE_BACK_COLOR -16777201 +
|MESSAGE_FONT_NAME MS Sans Serif +
|MESSAGE_FONT_SIZE 10|MESSAGE_BOLD ON +
|BUTTON_YES_CAPTION &ASCII +
|BUTTON_NO_CAPTION &BINARY +
|BUTTON_YES_COLOR -16777201 +
```

```
|BUTTON_NO_COLOR -16777201 +
|BUTTON_YES_FONT_COLOR 536870911 +
|BUTTON_NO_FONT_COLOR 536870911

-- prompt for the file name
PLUGINS LoadFileName.RBL vFileName +
|FULLPATH OFF +
|MULTISELECT OFF +
|TITLE Select file name to update

-- set a variable to the data in the file
SET VAR vFileData TEXT = ('[''' + .vFileName + ''']')

-- you must use an ampersand variable when the variable
-- contains the name of a file to be loaded
IF vFileType = 'YES' THEN
  UPDATE AppTable SET RawFileData = &vFileData WHERE RBFileName = .
vFileName
ELSE
  UPDATE AppTable SET BinFileData = &vfiledata WHERE RBFileName = .
vFileName
ENDIF
CLS
RETURN
```

Index

- S -

- T -

- U -

- V -

- W -

- X -

Notes