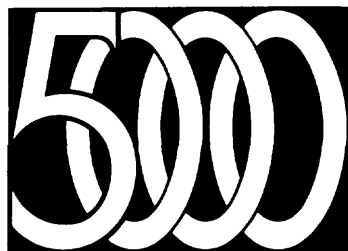


REFERENCE MANUAL

R:BASE
SERIES 5000
PC-DOS



B A S E



Copyright © 1985. All rights reserved.
By Microrim, Inc., Bellevue, Washington

PREFACE

How to Use This Manual

This Reference Manual has three sections:

- *R:base Command Dictionary*: The R:base commands described in alphabetical order with the command syntax, purpose, options, comments, and examples of how to use each command.
- *R:base 5000 Utilities*: Discussion of the other R:base 5000 main menu options: the EXPRESS, RBEDIT, RCOMPILE, and the FileGateway.
- *R:base 5000 Error Messages*: R:base error messages grouped by command plus error messages for the EXPRESS, RCOMPILE, and the FileGateway. The R:base messages include the error message number which is used for error trapping. The utility messages do not include the message number since error number trapping is not necessary when executing these functions.

Contents

R:BASE COMMAND DICTIONARY

How to Use This Chapter	R1-2
Command Dictionary Overview	R1-2
R:base Command Syntax Format	R1-2
Command Abbreviations	R1-4

R:BASE 5000 UTILITIES

How to Use This Chapter	R2-2
The EXPRESS	R2-3
RBEDIT	R2-4
RCOMPILE	R2-5
The FileGateway	R2-6

R:BASE 5000 ERROR MESSAGES

How to Use This Chapter	R3-2
R:base Messages	R3-3
General Processing Messages	R3-3
Column Messages	R3-4
Table Messages	R3-6
Relational Operation Messages	R3-6
SORT Clause Messages	R3-7
WHERE Clause Messages	R3-7
General Variable Messages	R3-8
Defining Variables	R3-8
Using Variables	R3-9
DOS and System Messages	R3-10
Command Specific Messages	R3-12
RCOMPILE Error Messages	R3-35
FileGateway Error Messages	R3-36
EXPRESS Error Messages	R3-42

R:base Command Dictionary

How to Use This Chapter

R1-2

Command Dictionary Overview

R1-2

R:base Command Syntax Format

R1-2

Command Abbreviations

R1-4

HOW TO USE THIS CHAPTER

Each command description can contain the following information. If a command does not require one of the listed topics, it is omitted from that command's description.

- **Syntax:** The command syntax and the meaning of keyword arguments
- **Purpose:** The purpose of the command
- **Options:** Description of Optional parts of the command
- **Comments:** Important implications of using the command
- **Examples:** Sample uses of the command

COMMAND DICTIONARY OVERVIEW

R:base Command Syntax Format

The general format for entering an R:base command is:

COMMAND KEYWORD argument KEYWORD argument . . .

Throughout this manual, commands and keywords are printed in upper case and arguments are printed in lower case. Optional portions of a command are offset from the command line. The command syntax is read from left to right with no breaks in the flow of the main line of the syntax. For example:

SYNTAX

```
graph LR; SELECT --- Choice1[ALL / collist]; Choice1 --- FROM[FROM tblname]; FROM --- Choice2[OPTIONAL: SORTED BY collist / WHERE condlist];
```

The diagram illustrates the syntax of the `SELECT` command. It shows the required parts of the command in a continuous flow: `SELECT`, followed by a choice between `ALL` and `collist`, then `FROM tblname`. Following this, there are two optional clauses: `SORTED BY collist` and `WHERE condlist`, which are shown in boxes that branch off from the main line of the syntax.

The word *SELECT* is required since it is on the main line. The words *ALL* and *collist* are choices—one of them must be used in the command so the left-right flow of the line is not broken. The words *FROM tblname* are required since they are on the main line. The two word groups *SORTED BY collist* and *WHERE condlist* are optional clauses which, if omitted, do not break the left-right flow of the syntax main line.

The keyword-argument combinations vary with each application of this command. In the example shown above, the COMMAND is `SELECT`, the KEYWORDS are `ALL`, `FROM`, `SORTED BY`, and `WHERE`, and the ARGUMENTS are *collist*, *tblname*, and *condlist*. The table and column names, and conditions must be provided by the user and be of the type shown in the syntax. For example, *tblname* indicates entry of a table name. The R:base reserved words (commands and keywords) do not change.

Table 1 describes the commonly used syntax arguments. Any unusual uses of the common arguments or arguments unique to a given command are explained following the syntax diagram for each command.

Table 1 Commonly Used Syntax Arguments

Argument	Meaning
=	Where no space is shown following an equal sign, then no space precedes the equal sign. Where an equal sign is shown with a space, then a matching space is on the opposite side.
chrname	Specified characters which may be set by the user. Including: BLANK, DOLLAR, QUOTES, SEMI, PLUS, and COMMA.
chrpos	Character position within a variable (see MOVE).
cmdfile	A file containing R:base commands. It has the same format as <i>filespec</i> .
colname	Identifies the name of a column and may be 1-8 characters long. Instead of typing the column name, you can enter <i>#c</i> where <i>c</i> is the item number shown when the columns are listed. Or, you can use a dotted variable whose value is the column name.
collist	A list of column names or <i>#c</i> . Or, you can use a dotted variable whose value is the column name.
condition	A comparison which defines specific rows to be acted upon by a command. Syntax for conditions are shown with the WHERE syntax.
condlist	A list of up to 10 conditions that identify the rows to be referenced. Used with WHERE, IF...THEN, and WHILE...ENDWHILE.
commandname	Used with HELP and PROMPT. A command available to either of these modes. Not all commands have on-line HELP or PROMPT capability.
datatype	A valid R:base datatype: DATE, DOLLAR, INTEGER, REAL, TEXT, or TIME.
dbspec	A database designation in the form, <i>d:\path\dbname</i> .
dbname	A 1-7 character database name.
d:	A drive letter.
else-block	Commands to be executed if the IF condition is not met (see IF...THEN).
expression	An arithmetic formula or value, or concatenated terms (see ASSIGN, SET VARIABLE).
filename.ext	A 1-8 character file name plus the optional a 1-3 character file extension <i>.ext</i> .
filespec	A unique DOS file specification in the form <i>d:\path\filename.ext</i> .
formname	Specifies the name of a table form and may be 1-8 characters long.
keylist	One or more special key designators used to indicate when to exit from an application. These keys are specified as ESC, ENTER, PGUP, and PGDN.
keyword	An R:base function used to determine the R:base environment. These include AUTOSKIP, BELL, CASE, CLEAR, COLOR, DATE, ECHO, ESCAPE, LINES, MESSAGE, NULL, REVERSE, RULES, SCRATCH, USER, and WIDTH. See SET and SHOW.
labelname	The name with LABEL to indicate where to skip to when a GOTO is executed.
length	The number of characters for a TEXT datatype column or variable.
listtype	LIST types are: DATABASES, TABLES, COLUMNS, RULES, FORMS, REPORTS, tblname.

Continued Next Page

Table 1 Commonly Used Syntax Arguments (*continued*)

Argument	Meaning
logicalop	Any of the operators: EQ, NE, LT, LE, GT, or GE.
menuname	Specifies the name of a menu and may be 1-8 characters long.
message	Message to be displayed on the screen.
newdbspec	New database specification.
#n	Specifies a route number where n is 1, 2, or 3 as determined by the SET POINTER command.
nchar	Number of characters (see MOVE).
paramlist	Parameter list. A list of up to nine values to be passed to a command file.
password	A 1-8 character OWNER, read, or write password.
path	A series of one or more directory names that lead from the root directory to a specified directory.
procfle	A binary procedure file created by RCOMPILE.
r	An integer designating number of rows.
(row,column)	Multiplan row and column used with UNLOAD.
rptname	Specifies the name of a report and may be 1-8 characters long.
scrnname	Specifies the name of a screen and may be 1-8 characters long.
scrnrow/col	The terminal screen rows and columns.
tblname	Specifies the name of a table and may be 1-8 characters long.
then-block	Commands to be executed if the IF condition is met (see IF ...THEN).
value	A constant amount usually to be entered into a column or variable.
varformname	A 1-8 character variable form name.
varname	Specifies a variable name and may be 1-8 characters long.
varlist	Specifies a list of variables.
= w	An equal sign followed by an integer specifying the display width for a column or variable.
while-block	Commands to be executed if the WHILE condition is met (see WHILE...ENDWHILE).

*If syntax uses more than one argument, each argument is numbered—*argument1*, *argument2*, and so on. For example, *tblname1*, *colname2*, *chrpos 2*.

All commands are executed by pressing the [ENTER] key at the end of the entire command entry.

A command may be repeated by pressing the [Ctrl] and right-arrow keys simultaneously. See chapter 5 for a complete explanation of command editing.

If a command is incorrectly entered, R:base responds with an error message and a help screen that displays the correct syntax for that command if R:base can identify the command you are attempting to use.

Command Abbreviations

Commands and keywords that are comprised of four or more letters can be abbreviated to as few as three letters. For example, INT can be used for INTERSECT. If a command or keyword's first three letters are not unique (for example, ENDWHILE and ENDIF), the abbreviation must be at least long enough to be unique (for example, ENDW and ENDI).

APPEND

SYNTAX

```
APPEND tblname1 TO tblname2 WHERE condlist
```

tblname1 is the source table (the one to be appended)
tblname2 is the destination table (the one to which *tblname1* is appended)

- Purpose** APPEND copies rows from one table (the source) to the end of another table (the destination).
- Options** You can use a **WHERE** clause with this command to limit the number of rows affected by the APPEND command.
- Comments** Only the columns in the first table that are common to the second table are used. Columns in the source table that are not in the destination are not used. Columns in the destination table that are not in the source table are filled with nulls.

Examples APPEND newemp TO emtable

Figure 1 shows the results of this example.

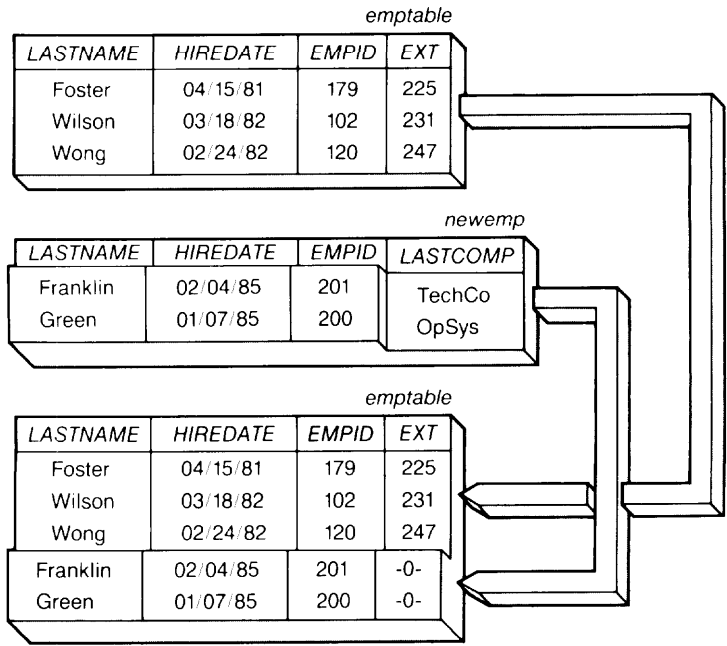


Figure 1 An Example of the APPEND Command

ASSIGN

SYNTAX

```

ASSIGN colname TO expression IN tblname

    WHERE condlist
  
  #n
  
```

expression is an arithmetic calculation or a concatenation process used to obtain a value

Purpose **ASSIGN** is a data modification command that changes the data value of a column to the result of a numeric calculation or string concatenation.

Options *WHERE...*: A clause used to limit the rows affected by the **ASSIGN** command.

IN #n: A route number (n equals 1, 2, or 3) which points to a specific row to be affected by the **ASSIGN** command. The route is established with the **SET POINTER** command.

Comments Use a **WHERE** clause or a route number to pinpoint the column values you want to change.

Expressions are in any of the following forms:

```

colname op colname
colname op value
value    op colname
value    op value
  
```

Table 2 lists the operators (op).

Table 2 Operators Used with the ASSIGN Command

Operator	Function	Example
+	Adds the two items, or concatenates two literal terms	12 + 13 = 25 jo + in = join
&	Concatenates two literal terms but leaves one space between	Sam & Evans = Sam Evans
-	Subtracts the second item from the first	25 - 10 = 15
/	Divides the first item by the second	16 / 2 = 8
X	Multiplies the two items	8 x 2 = 16
%	The first item is the percentage value of the second item*	50 % 16 = 8

*Percentage is calculated as (item1 x item2) / 100. If both items are INTEGER, the result is INTEGER. If one or both items are REAL, the result is REAL.

Operators and the equal sign (=) must be preceded and followed by a space as shown in the examples in table 2.

Expressions with DOLLAR data types yield the following results:

DOLLAR + DOLLAR = DOLLAR	DOLLAR X REAL = DOLLAR
DOLLAR - DOLLAR = DOLLAR	DOLLAR / REAL = DOLLAR
DOLLAR / DOLLAR = REAL	REAL X DOLLAR = DOLLAR
DOLLAR X INTEGER = DOLLAR	REAL % DOLLAR = DOLLAR
DOLLAR / INTEGER = DOLLAR	INTEGER X DOLLAR = DOLLAR
	INTEGER % DOLLAR = DOLLAR

Expressions with DATE data types yield the following results:

DATE - DATE = day	DATE + INTEGER = newdate
DATE - INTEGER = newdate	

Expressions with TIME data types yield the following results:

TIME - TIME = seconds	TIME + INTEGER = newtime
TIME - INTEGER = newtime	

Expressions that calculate the difference between two dates (days) or times (seconds) result in INTEGER values. Make sure that the column assigned to such an expression is an INTEGER data type.

When an expression is computed, values that exceed the limits of a column or variable are assigned null values (column limits are determined when the column is defined). If there is not enough room to display the value, an asterisk (*) is displayed.

Any expressions that contain a null value produce a null result.

Global variables can be used in expressions. Precede the variable name with a period in the expression (for example, *.variable*).

Examples

ASSIGN costs TO 1.1 x costs IN cashflow

Assigns the value of the expression *1.1 x costs* to the column *costs* for every row in the table *cashflow*.

ASSIGN units TO units + quantity IN transx WHERE transid LT 5000

Assigns the value of the expression *units + quantity* to the column *units* in the table *transx* for rows where *transid* is less than 5000.

ASSIGN onhand TO onhand - quantity IN #3

Assigns the value of the expression *onhand - quantity* to the column *onhand* in table *product* in the row number designated in row pointer 3.

ASSIGN onhand TO onhand - .var1 IN product WHERE COUNT EQ 5

Uses the value of the global variable *var1* in the expression that is assigned to the *onhand* column located in the fifth row.

ASSIGN fullname TO firstname & lastname IN custlist

Concatenates the TEXT columns *firstname* and *lastname*, leaving one space between values, and stores the results in the *fullname* column in every row in the table *custlist*.

BREAK

SYNTAX

BREAK

Purpose	BREAK is used within command files, only within a WHILE loop. It enables an early exit from the loop.
Comments	The BREAK command is executed in an IF ... ENDIF structure contained in the WHILE ... ENDWHILE loop. The IF conditions indicate when to execute the BREAK command.
Example	See WHILE ... ENDWHILE for examples of using the BREAK command.

BUILD KEY

SYNTAX

BUILD KEY FOR colname IN tblname

- Purpose** BUILD KEY is a schema modification command that creates a key index for a column. A key provides a direct pointer to the location of the row.
- Comments** R:base uses keys to find a requested column more quickly when it is specified in a WHERE clause. To take advantage of key processing, WHERE clauses must conform to the following conditions:
- The last condition in the clause must refer to a key column
 - The last operator used in the condition must be EQ or =
 - If more than one condition is specified in the clause, the last condition must be preceded by the logical operand AND

CHANGE Values

SYNTAX

```
CHANGE colname TO value [IN tblname] [WHERE condition]
                        [IN #n]
```

value is the new data value or global variable name

Purpose *CHANGE* is a data modification command that changes the value of a column in a single table or in all tables in a database.

Options *WHERE...*: A clause used to limit the rows affected by the *CHANGE* command.

IN tblname: Lets you specify a table where the column value is changed.

IN #n: A route number (where *n* is 1, 2, or 3) which points to a specific row to be affected by the *CHANGE* command. The route is established with the *SET POINTER* command.

Comments To change the value of a column in a single table, specify a table name.

To change the value of a column in all tables where that column appears, leave out the table name.

You must always use a *WHERE* clause or a row pointer to ensure that you do not inadvertently change the values in more rows than you intend.

Do not use wildcard characters (*) in the value. R:base assumes the value is literal and does not accept wildcard values.

Examples *CHANGE custid TO 109 IN transx WHERE transid = 4099*

Changes the value of the customer id to 109 in the table *transx* if the transaction id number is equal to 4099.

CHANGE custid TO .var1 IN transx WHERE address EQ "3 N. Elm"

Changes the customer id to the current value of *var1* for all rows in the table where the customer address is 3 N. Elm.

CHECK/NOCHECK

To turn on rule checking when you are in the LOAD mode, type

```
SYNTAX
```

```
CHECK
```

To turn off rule checking when you are in the LOAD mode, type

```
SYNTAX
```

```
NOCHECK
```

Purpose	When R:base is in the LOAD mode (L>), CHECK directs R:base to check input against all existing rules and NOCHECK directs R:base to ignore all existing rules. The R:base default value is CHECK.
Comments	<p>The CHECK and NOCHECK commands are used only when loading data and are only recognized when entered in LOAD mode (at an L.> prompt).</p> <p>If you have defined an owner password for your database, R:base does not accept the NOCHECK command until you enter the password.</p> <p>The CHECK and NOCHECK commands work independently of the SET RULES ON/OFF commands. If rules are set off, you can still use the CHECK command to verify data entries with rules.</p>

Examples `CHANGE COLUMN transid IN transx TO text 20`

Changes the data type of column *transid* to TEXT with a length of 20 characters. The change is effective only in table *transx*.

`CHANGE COLUMN company IN custlist TO custname`

Changes the name of column *company* to *custname* in table *custlist*.

CHANGE COLUMN

SYNTAX

```
CHANGE COLUMN colname1 IN tblname TO {
  datatype      length
  colname2
  colname2 datatype length
}
```

colname2 is the new column name

datatype is the new data type

length is the new column length

Purpose The **CHANGE COLUMN** command enables you to completely redefine a column. The column name, type, and length can all be changed, or individually modified. Column lengths can only be modified for TEXT data types.

Options *colname2* is the new column name if the name is to be changed.

datatype need only be entered if the data type of the column is changed.

length can only be entered if the column's new data type is TEXT.

Comments The column change is limited to a single table.

A temporary table is created by this command so you must have at least one table (no more than 39 tables existing) and one column (no more than 399 columns existing) available.

If the column name exists in more than one table, you must specify a new column name to use the **CHANGE COLUMN** command. Alternatively, you can **RENAME** the existing column.

The command keywords can be either **CHANGE COL** or **CHANGE COLUMN**.

The column description can be changed in any of the following combinations:

- Column name only
- Column name and data type
- Column name, data type, and length
- Data type only
- Data type and length only

CHDIR

SYNTAX

CHDIR d: path

d: is a designator for an available drive

path is a series of one or more directory names that lead from the root directory to a specified subdirectory

Purpose The CHDIR (change directory) command lets you change the current directory on either the default drive or a drive that you specify. CHDIR, entered without parameters, allows you to view the default drive and the current directory.

Options You can use the short form CD for CHDIR.

Comments Whenever the R> prompt is displayed, you can change directories. Include a space after you enter CHDIR or CD. R:base assumes you want to work in the same directory and drive as those of the open database until you specify a change with this command.

This command is similar to the DOS CHDIR command. See your DOS manual for information on messages.

Example CHDIR c:\salesdept

Executing this command changes the default drive to *c:* and the current directory to *salesdept*.

CHDRV

SYNTAX

`CHDRV` *d*:

d: is a designator for an available drive

- Purpose** CHDRV either displays the default drive designator or changes the default drive to another drive that you specify.
- Options** To change drives, you may omit the command name and enter the drive specification alone, as you do in DOS.
- Comments** Whenever the R> prompt is displayed, you can use this command. To display the default drive, enter the command only. R:base will display the drive designator. To change drives, enter the command, specifying the new drive. R:base and DOS assume you want to use the default drive until you specify a change with this command.
- Examples** CHDRV
- If the default drive is *a*:, for example, R:base will display the drive designator like this:
- A:
- CHDRV c:
- Executing this command changes the default drive to *c*:.
- C:
- Entering this drive letter at the R> prompt changes the default drive to *c*: if you were on another drive.

CHKDSK

SYNTAX

CHKDSK

d:

d: is a designator for an available drive

Purpose The CHKDSK (check disk) command shows the total number of bytes of disk space and memory as well as the disk space and memory available for use.

Options Specify the drive if you want to check the status of a disk other than the default drive.

Examples CHKDSK a:

Executing this command provides a display similar to this:

```
362496 bytes total disk space
98304 bytes available on disk

393216 bytes available
76536 bytes free
```

CHOOSE

SYNTAX

```
CHOOSE varname FROM menuname
                                IN procfile
```

varname is the variable that will hold the value entered (the menu selection)

menuname is the menu to be displayed

procfile is the procedure file that contains the menu

Purpose The CHOOSE command causes a specified R:base 5000 horizontal or vertical menu to be displayed, waits for a menu selection to be entered from the keyboard, and stores the keyboard input in a designated global variable which can be referenced in subsequent processing.

Options The menu may be stored either in a DOS file or as a menu block in an R:base procedure file created by RCOMPILE. If the *procfile* option is used, *menuname* refers to a menu in *procfile*. If the *procfile* option is not used, then R:base searches the last used procedure file for the menu. If the menu is not found in that procedure file, or no procedure file was previously used, then R:base uses *menuname* as the name of a DOS file that contains the menu to be displayed.

In the *procfile* option, a drive letter is needed if the menu is not stored on the default drive, and a path is needed if the menu is not in the current directory on the drive.

Comments This command displays two menu types: horizontal or vertical. A horizontal menu is displayed horizontally across the screen and can contain up to 40 menu options. The text for each option can contain, at most, 12 characters. The operator selects an option by highlighting the option text and pressing [ENTER].

A vertical menu displays a list of options, each of which is associated with an option number. The operator selects an option by highlighting the option number and pressing [ENTER]. A vertical menu can contain, at most, nine choices. The text that describes a choice can contain up to 65 characters.

The total number of text characters in all the choices of a menu must not exceed $(400 - (2*N))$ characters, where N is the number of menu options.

Examples CHOOSE choice FROM mainmenu IN compproc

Searches the procedure file *compproc* for the menu *mainmenu*, displays the menu on the screen, and waits for input from the keyboard. When the operator enters a selection, it is stored in the global variable *choice*.

CHOOSE choice FROM mainmenu IN \newdir\compproc

Same as above, except the procedure file *compproc* is not in the current directory.

CHOOSE choice FROM mainmenu

Same as above, except if *mainmenu* is not found in the procedure file last used by the command file, then the menu contained in the file *mainmenu* is displayed.

CLEAR

SYNTAX

```
CLEAR {  
    varname  
    ALL VARIABLES  
}
```

- Purpose** The CLEAR command is used to remove a single global variable or all global variables from memory. Typically, CLEAR is used in command files.
- Options** Use the *varname* option to remove a specific variable. Use this option in one or more CLEAR commands at the end of a complete set of procedures to clear unnecessary variables. This is good housekeeping and recommended.
- Use the ALL VARIABLES option to remove all global variables from memory.
- Comments** When R:base is first loaded into memory, no variables are defined.
- Examples** CLEAR counter
- Removes the global variable *counter* from memory.
- CLEAR ALL VARIABLES or CLE ALL VAR
- Removes all global variables from memory.

CLOSE

SYNTAX

CLOSE

Purpose	The CLOSE command writes any modifications of the currently open database to disk and then closes the database.
Comments	The EXIT or OPEN commands also close a currently open database. Use the CLOSE command when the next database you want to open is on a different floppy disk than the currently open database or when you want to leave the open database without opening another.

COLUMNS and Column Definitions

SYNTAX

COLUMNS

The column definitions, which are entered after the COLUMNS subcommand executes, have this format:

SYNTAX

```
colname datatype [length] [KEY]
```

datatype specifies the data type of the new column

length is an integer that specifies the maximum number of characters allowed for items in TEXT columns (unless you are defining an array)

key specifies that an index entry is to be built for *colname*

Purpose The COLUMNS command is used in conjunction with the DEFINE command to start the column definition process.

Comments Within R:base, columns are defined in the DEFINE or EXPAND modes.

Each column definition has up to four parts: a name, data type, optional length, and an optional key designation. Enter column item. Length is only used with TEXT data types unless you are defining an array. An array is created by specifying a length for INTEGER or REAL data type columns. If the length is not entered for a TEXT column, it defaults to eight characters. TEXT column lengths can range from 1 to 1500 characters.

Column names may contain from 1 to 8 characters.

Valid data types are DATE, DOLLAR, INTEGER, REAL, TEXT, and TIME.

If you define columns then do not use the TABLE command to place the columns into a table, the column definitions are lost when you exit from DEFINE mode with the END command.

Examples COLUMNS

Enter the COLUMNS command only when in the *DEFINE* mode at a D> prompt. R:base then expects the column definitions to be entered in the appropriate format. After you finish entering column definitions, use the TABLES command to group the columns together in tables.

```
custname TEXT 20  
custid INTEGER KEY
```

The column definitions above are typical entries. The first entry defines a TEXT column named *custname* and specifies a maximum length of 20 characters. The second entry defines an INTEGER column named *custid* and specifies the column as a key.

COMMENT

SYNTAX

*(comment text)

comment text is any string of characters

Purpose	Comments may be included in a command file but are not executed when the file is run. They are remarks about the command file.
Options	A comment may share a command line with a command, occupy a line itself, or extend over multiple command lines. See the example of each below.
Comments	<p>Although used primarily in command files, a comment can be entered when using R:base interactively.</p> <p>When R:base encounters *(, the text following is interpreted as a comment until a matching parenthesis is reached. For example,</p> <p>*(THIS IS (123) A VALID COMMENT)</p> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;"> ↑ internal parenthesis </div> <div style="text-align: center;"> ↑ closing parenthesis for first </div> </div>
Examples	<p>*(This is a comment that occupies one line)</p> <p>*(This is a comment that occupies more than one line)</p> <p>SET VAR count TO 1 *(Initialize the variable named count)</p> <p>SET VAR count1 TO 1 ; SET VAR count2 TO 1 *(Initialize two variables)</p>

COMPUTE

SYNTAX

```

COMPUTE [varname AS] [
  AVE
  COUNT
  MAX
  MIN
  SUM
] colname FROM tblname [WHERE condlist]

COMPUTE [varname AS] ROWS FROM tblname

COMPUTE ALL colname FROM tblname [WHERE condlist]

```

- Purpose** The **COMPUTE** command performs calculations on selected columns. You can determine averages and totals for **DOLLAR**, **INTEGER**, and **REAL** columns. You can compute minimums, maximums, and count for all data types.
- Options** *WHERE*: This clause is used to limit or select the rows on which the **COMPUTE** is performed.
- varname AS*: The result of each computation can either be displayed or stored in a global variable. The variable must be **INTEGER** data type if **COUNT** or **ROWS** is used.
- Comments** Table 3 lists the **COMPUTE** command arguments.

Table 3 COMPUTE Command Arguments

Command	Function
ALL	Computes all six of the following mathematical operations.
AVE*	Computes the arithmetic average of DOLLAR, INTEGER, or REAL data types. Averages of integer values are rounded to the nearest integer value and dollars are rounded to pennies.
COUNT	Determines how many entries there are for a particular column item.
MAX*	Selects the row in a column with the greatest arithmetic or alphabetic value. For TEXT data types, the first 20 characters are evaluated.
MIN*	Selects the row in a column with the least arithmetic or alphabetic value. For TEXT data types, the first 20 characters are evaluated.
ROWS	Determines how many rows there are for a specific column.
SUM*	Computes the arithmetic sum of DOLLAR, INTEGER, or REAL data types.
*When these functions are performed on INTEGER or REAL arrays, then only the first value is used.	

Examples COMPUTE AVE units FROM transx WHERE custid = 101

Computes the average for the *units* sold in table *transx* if the customer id equals 101.

COMPUTE minvar AS MIN onhand FROM product

Stores the minimum on-hand quantity in the variable *minvar* as found in table *product*.

COMPUTE rowctr AS ROWS FROM transx

Computes the total number of rows in the table *transx* and places the result in variable *rowctr*.

COPY

SYNTAX

COPY filespec1 filespec2

filespec1 is the DOS source file

filespec2 is the DOS target file

- Purpose** The COPY command lets you copy one or more DOS files to another drive, to another directory, or to the same directory.
- Options** You can use the global filename characters ? and *. The ? character can be used in place of any one character in a filename. The * character can be used in place of a single character and the characters that follow.
- Comments** This command is similar to the DOS COPY command.
- If you copy from one drive to another, include the drive designation for any drive other than your default drive. If you copy from one directory to another, include the name of the directory for any directory other than the current directory.
- If you copy from one drive or directory to another, you can keep the same name for the new file, or you can give it a new name. If you make a copy on the same directory, you must give the new file a new name.
- Examples** COPY mydata b:
- Executing this command copies the file *mydata* from the default drive and current directory to the default directory of drive *b:*.
- COPY \dbs\mydata*.rbs \admin
- Executing this command copies any files with names that begin with *mydata* and have the extension *rbs*, such as the three DOS files that hold an R:base database—*mydata1.rbs*, *mydata2.rbs*, and *mydata3.rbs*. The new files are copied to the directory *admin*.

DEFINE

SYNTAX

```
DEFINE [dbspec]
```

Purpose	The DEFINE mode enables you to specify all of the characteristics for a specific database. The characteristics that must be defined are the database name, columns, and tables. Optional characteristics include passwords and rules.
Options	<p>If you want to redefine the characteristics of the open database, the database name may be omitted.</p> <p>The database you specify may be a new or an existing database.</p> <p>If you want to define the database on a drive or directory different from your current directory, include the drive and subdirectory specification before the database name.</p>
Comments	<p>When you enter the DEFINE command for a new database, you have to include a database name.</p> <p>Database names can be no longer than seven characters and may not contain blanks or a file extension. Drive and subdirectory specifications are not included in the seven character limit. R:base automatically adds the extension <i>.RBS</i> to the database file names when they are created.</p>
Examples	<p>DEFINE</p> <p>This command enters the DEFINE mode for the currently open database.</p> <p>DEFINE c:\database\compuco</p> <p>This command opens a database named <i>compuco</i> in the <i>database</i> subdirectory of your <i>c:</i> drive.</p>

DELETE

SYNTAX

```
DELETE ROWS FROM tblname WHERE condlist
                  |
                  |#n
or
DELETE DUPLICATES FROM tblname
or
DELETE KEY FOR colname IN tblname
```

Purpose	The DELETE command deletes selected rows, duplicate rows, or a key designation from a table.
Options	<p>#n: A route number (n equals 1, 2, or 3) which points to a specific row to be affected by the DELETE command. The route is established with the SET POINTER command.</p> <p>WHERE ...: The WHERE clause is required if a table name is specified. To delete all rows from the table, indicate a condition which must be true for all rows. For example, WHERE <i>colname</i> EXISTS.</p>
Comments	<p>To delete selected or all rows from a table, enter the DELETE ROWS command.</p> <p>To delete duplicate rows from a table, enter the DELETE DUPLICATES command. A duplicate row is defined as a row where the values for each column are exactly the same as another row in the table.</p> <p>The DELETE DUPLICATES option processes much faster if the row contains a key column.</p> <p>The DELETE KEY command removes a key which may be reestablished by using the BUILD KEY command.</p>
Examples	<p>DELETE DUPLICATES FROM transx</p> <p>Deletes the duplicate rows from the table <i>transx</i>.</p> <p>DELETE ROWS FROM transx WHERE custid = 100</p> <p>Deletes the rows in table <i>transx</i> where the <i>custid</i> value is 100.</p> <p>DELETE KEY FOR custid IN transx</p> <p>Removes the key designation from <i>custid</i> in the table <i>transx</i>.</p>

DIR

SYNTAX

DIR

filespec

Purpose	The DIR (directory) command lists the files on a directory.
Comments	<p>If you enter only <i>DIR</i>, R:base displays the current directory. You can add a drive designation and path to display files for another directory.</p> <p>The directory display includes an identification of the directory, the names, sizes, dates and times of the last updates, total number of files, and the space available on the disk.</p> <p>If you specify the filename, DOS displays information for that file only. You can also use the wildcard characters * or ? to display a group of files.</p> <p>This command is similar to the DOS DIR command. The R:base DIR command, however, pauses at the end of each full screen of information, and it does not support the wide-display (/w) option of the DOS DIR command.</p> <p>See your DOS manual for information on messages.</p>
Examples	<p>DIR c:\mrktng</p> <p>Executing this command displays a list of files from the directory <i>mrktng</i> on drive <i>c:</i>.</p> <p>DIR mydb*.*</p> <p>Executing this command displays information from the current directory about all files whose names begin <i>mydb</i>—for example, the three files that hold an R:base database: <i>mydb1.rbs</i>, <i>mydb2.rbs</i>, and <i>mydb3.rbs</i>.</p>

DISPLAY

SYNTAX

```
DISPLAY scrnname                       
                    IN procfile           
```

scrnname refers to a screen block stored in an application or procedure file or to any DOS file whose contents are to be displayed

procfile refers to a procedure file

- | | |
|----------|--|
| Purpose | The DISPLAY command, used in a command file, displays a stored screen. |
| Options | <p>If the <i>procfile</i> option is not used, then R:base will search for the screen name in the procedure file that was most recently used by the command file that contains the DISPLAY command.</p> <p>If the command file has not used a procedure file or the screen name cannot be found in the most recently used procedure file, then <i>scrnname</i> is used as the filename that contains the stored screen. In that case, the DISPLAY command is functionally equivalent to the TYPE command.</p> |
| Comments | The DISPLAY command can display text that explicitly tells the operator how to make a menu choice. To allow entry, a FILLIN command must be used to accept the operator's response. |
| Examples | <p>DISPLAY screen1 IN b:myproc.prc</p> <p>Displays the screen named <i>screen1</i> which is in the procedure file <i>myproc.prc</i> residing on drive <i>b:</i>.</p> <p>DISPLAY mainmenu ; FILLIN v1 USING "Enter a letter (A - F)" AT 16,20</p> <p>Displays the menu text, which is a file named <i>mainmenu</i> in the current directory on the default drive. This command line also writes the operator prompt starting at row 16, column 20 on the screen.</p> |

DRAW

SYNTAX

DRAW *varformname*

WITH VARIABLE *varlist*

AT *scrnrow*

WITH ALL

varformname is a stored variable form (see FORMS)

varlist is a list of variables

scrnrow is an integer between 2 and 24 that specifies the screen line at which the top of the variable form is positioned

Purpose The DRAW command, used in a command file, is used to display the text of a variable form. The current value of global variables used in the form may also be displayed with the text.

Options The command without options will display the variable form text without displaying the current value of any of the variables defined for the variable form.

WITH VARIABLE or *WITH ALL*: This option is used to include one or more variable values in the display. The *WITH ALL* option is used to display values for all of the variables defined for the variable form. Use the *WITH VARIABLE varlist* option to limit the variables the operator can view.

AT scrnrow: Use this option to position the top of the form at any line on the screen other than the top or bottom lines (lines 1 and 25). If this option is not used, the top of the form is drawn on line 2.

Line 1 of the screen is always reserved as the operator prompt line. Line 25 is for error messages.

Comments As many as five variable forms can be displayed at one time on the screen by executing DRAW commands without clearing the screen with a NEWPAGE command. After a variable form has been drawn, an ENTER VARIABLE command can be executed to give the operator an opportunity to enter values into the variable fields of the form. Or, an EDIT VARIABLE command can be executed to give the operator the opportunity to change variable values that are displayed with the form.

The DRAW command can be used to define global variables (the same function as performed by SET VAR, FILLIN, and COMPUTE). Any variables named in a variable form which are not currently defined are defined when the form is drawn. The newly-defined variables, named explicitly in the WITH VARIABLE option or implicitly in the WITH ALL option, will be displayed as null.

Examples

DRAW vf1

Draws the formatted text of the variable form *vf1* on the screen. No variables are included in the display. The top of the form is positioned at line 2 on the screen.

DRAW vf1 WITH ALL

Draws the formatted text of the variable form *vf1* on the screen. The values of all the variables defined in *vf1* are included in the display. The top of the form is positioned at line 2 on the screen.

DRAW vf1 WITH VARIABLES custfx custfy custfz AT 10

Draws the formatted text of the variable form *vf1* on the screen. The values of the variables *custfx*, *custfy*, and *custfz* are included in the display. The top of the form is positioned at line 10 on the screen.

DRAW vf1 WITH ALL ; DRAW vf2 WITH ALL AT 9 ; DRAW vf3 WITH ALL + AT 16

Draws the formatted text of three variable forms (*vf1*, *vf2*, and *vf3*) on the screen. The top of *vf1* is positioned at line 2, the top of *vf2* is positioned at line 9, and the top of *vf3* is positioned at line 16. The current values of all the variables defined for each of the three forms are also displayed.

EDIT

SYNTAX



- Purpose** The **EDIT** command is used to browse up or down through the rows and columns of a table, change data, or delete rows.
- Options** **=w**: Selects column display widths for the named columns. Enter the width after each column name.
- SORTED BY...**: This clause sorts the rows by the column(s) specified in the sort clause.
- WHERE...**: This clause is used to select the rows to be edited.
- Comments** You can move up, down, right, or left in the table. R:base allows you to scroll forward until you reach the end of the file or backwards through the last fifty rows.
- If the table is wider than the screen, R:base indicates that there is more data and specifies which direction it lies by displaying the word *more* and an arrow at the top of the screen. For example:
- ← more or more →
- When you edit data, R:base shows that you are changing information by displaying the word *update* at the top of the screen. After you modify the entry and move the cursor to another item in the table, the change is recorded and the update message is cancelled. The changes that you make are not sent to the disk until editing is complete and you press the [ESC] key.
- Refer to chapter 9 in the *R:base 5000 User's Manual* for a list of function keys used with the **EDIT** command.
- Examples** **EDIT ALL FROM transx SORTED BY custid**
- Displays each row for editing from table *transx* in customer id order.
- EDIT custid company=10 cname cname FROM custlist WHERE custid = 100**
- Displays only the rows in table *custlist* where the customer id is equal to 100. The only columns displayed for editing are *custid*, *company*, *cname*, and *cname*. The display for *company* is 10 characters wide.

EDIT USING

SYNTAX

```
EDIT USING formname [SORTED BY collist] [WHERE condlist]
```

Purpose	The EDIT USING command is used to change or update data, or delete rows that are displayed in the format of the specified table form.
Options	<p>SORTED BY...: This clause sorts the rows by the column(s) you specify in the sort clause.</p> <p>WHERE...: This clause is used to limit the rows to be edited.</p>
Comments	<p>This command works with data in table forms. Data in variable forms is edited with the EDIT VARIABLE command.</p> <p>This command displays the data on the screen one row at a time using a previously created table form (see FORMS for instructions on how to set up a table form).</p>
Examples	<p>EDIT USING tranform SORTED BY custid</p> <p>Edits the table specified for form <i>tranform</i> and displays the rows in customer id order.</p> <p>EDIT USING transform WHERE custid = 100</p> <p>Edits the table specified for form <i>tranform</i> and displays only the rows in which the customer id is equal to 100.</p>

EDIT VARIABLE

SYNTAX

```
EDIT VARIABLE [varlist] [USING varformname] [RETURN keylist]
```

varlist is a list of variables

varformname is a stored variable form (see FORMS)

keylist is a list of keyboard keys used for operator interaction: ESC, ENTER, PGUP, PGDN

Purpose

The EDIT VARIABLE command enables an operator to edit information using a predefined variable form. The current variable values may be displayed on the screen embedded in variable form text, which is drawn on the screen by a DRAW command. If no DRAW command is executed before the EDIT VARIABLE command, the *USING varformname* clause must be included and the values displayed without supporting text.

Options

varlist: Use this option to limit the variables that are used for editing data.

USING varformname: If more than one variable form is drawn on the screen, this option can be used to identify the form within which the editing is to take place (which form is to be the active form). See the DRAW command description for information about displaying more than one form at a time. It must also be used when no DRAW command is used before the EDIT VARIABLE command.

RETURN keylist: This option indicates the names of keys the operator will press to signal that editing has been completed within the active form. Key names that can be used in the list are ESC, ENTER, PGUP, and PGDN. A keylist may contain from one to four of these key names, in any order. These different signals can be interpreted by the application to decide what to do next because the name of the key most recently pressed by the operator is always available to the application in the system variable #RETURN.

Comments If you want to edit table data, use the EDIT or EDIT USING commands.

The [ENTER] key must be pressed after changing the value in a field. If you have not changed the value (the cursor is positioned at the beginning of the entry field), pressing [ENTER] moves the cursor to the next field without changing the value of the last field.

The EDIT VARIABLE command does variable data type checking of all new values entered by the operator. If the format or content of the entry does not match the data type of the defined variable, and the R:base environment contains SET ERROR MESSAGES OFF, EDIT VARIABLE fills that variable with a null value rather than the entered value. If the R:base environment contains SET ERROR MESSAGES ON, EDIT VARIABLE displays an error message Data in this field must be screen type TTTT, where TTTT is TEXT, DATE, TIME, INTEGER, REAL, or DOLLAR. The EDIT VAR form-field editor will not move the data entry cursor to the next field in the form until the operator enters a null value or a value that matches the data type displayed in the error message.

Examples EDIT VARIABLE

Gives the operator cursor access to all the displayed variable values in the active variable form for purposes of changing one or more of the values.

EDIT VARIABLE salary

Gives the operator cursor access to one of the displayed variable values in the active variable form, namely variable *salary*. If other variable values are displayed on the form, the operator will have no opportunity to change them.

EDIT VARIABLE salary USING vf2

Gives the operator cursor access to the variable *salary*, but access to no other variable, in the variable form named *vf2*, whether or not *vf2* is the active form. If there is no active form, the current value of *salary* is displayed on the screen without any form text (in other words, an EDIT VARIABLE command can be executed without a DRAW command being executed first).

EDIT VAR USING v/2 RETURN ESC PGUP PGDN

Gives the operator cursor access to all the displayed variable values in the variable form named *v/2* for purposes of changing one or more values. The operator can signal that all changes have been made by pressing any one of the three keys [ESC], [PgUp], or [PgDn]. The application could interpret [PgUp] to mean "Draw the previous part of a multi-part form next," [PgDn] to mean "Draw the next part of a multi-part form next," and [ESC] to mean "Erase the screen because all parts of a multi-part form are edited." The application can determine which key the operator pressed most recently by interrogating the value of the #RETURN system variable.

ENTER

SYNTAX

```
ENTER formname [ FROM filespec  
                FOR r ROWS ]
```

- Purpose** The ENTER command adds data to a table using a previously defined table form (see FORMS).
- Options** *FROM filespec*: This clause indicates that the data is entered from an external ASCII file rather than from keyboard entry.
- FOR r ROWS*: This clause limits the number of rows entered to the integer number represented by *r*.
- Comments** If adding data to a table via the keyboard, omit the file specification. If adding data from an ASCII fixed field data file with rows less than or equal to 80 characters, define a form that matches column entry locations to file locations.
- Examples** ENTER tranform FROM b:\transact\trans.dat
- Uses the form *tranform* to load data from the external file *trans.dat* residing on drive *b*: in subdirectory *transact*. See chapter 12 for more information.
- ENTER tranform FOR 1 ROW
- Uses form *tranform* to enter one row of data via the keyboard.

ENTER VARIABLE

SYNTAX

```
ENTER VARIABLE [varlist] [USING varformname] [RETURN keylist]
```

varlist is a list of the variables form

varformname is a stored variable form (see FORMS)

keylist is a list of keyboard keys used for operator interaction: ESC, ENTER, PGUP, and PGDN

Purpose The ENTER VARIABLE command enables an operator to enter information using a predefined variable form. Screen areas within which to enter each value appear on the screen embedded in variable form text, which is drawn on the screen by a DRAW command. If no DRAW command is executed before the ENTER VARIABLE command, you must use a USING clause, and then the entry fields appear without supporting text.

Options *varlist*: Use this option to limit the variables that are used for entering data.

USING varformname: If more than one variable form is drawn on the screen, this option can be used to identify the form within which the entry is to take place (which form is to be the active form). See the DRAW command description for information about displaying more than one form at a time. If you want to enter variables without first drawing the variable form, you must use the USING clause.

RETURN keylist: This option indicates the key names the operator will use to signal that entry has been completed within the active form. Key names that can be used in the list are ESC, ENTER, PGUP, and PGDN. A keylist may contain from one to four of these key names, in any order. These different signals can be interpreted by the application to decide what to do next because the name of the key most recently pressed by the operator is always available to the application in the system variable #RETURN.

Comments	<p>To enter data with a table form, use the <i>ENTER formname</i> command.</p> <p>Pressing the [ENTER] key alone in response to a prompt sets the value to null.</p> <p>The ENTER VARIABLE command does variable data type checking on all new values entered by the operator. If the format or content of the entry does not match the data type of the defined variable, and the R:base environment contains SET ERROR MESSAGES OFF, ENTER VARIABLE fills that variable with a null value rather than the entered value. If the R:base environment contains SET ERROR MESSAGES ON, ENTER VARIABLE displays an error message Data in this field must be screen type TTTT, where TTTT is TEXT, DATE, TIME, INTEGER, REAL, or DOLLAR. The ENTER VARIABLE editor will not move the data entry cursor to the next field in the form until the operator enters a null value or a value that matches the data type displayed in the error message.</p> <p>If you use the SET MESSAGES OFF command, the default prompt Press ESC when done with this data is not displayed. However, you can display your own custom prompt on the line using a WRITE command.</p>
Examples	<p>ENTER VARIABLE</p> <p>Gives the operator cursor access to all the variable fields in the active variable form so values can be entered.</p> <p>ENTER VARIABLE salary</p> <p>Gives the operator cursor access to one of the fields in the active variable form, namely, the field for the variable <i>salary</i>. If other variable fields are displayed on the form, the operator will have no opportunity to enter values into them.</p> <p>ENTER VARIABLE salary USING vf2</p> <p>Gives the operator cursor access to the field for the variable <i>salary</i>, but access to no other field, in the drawn variable form named <i>vf2</i>, whether or not <i>vf2</i> is the active form. If <i>vf2</i> is not previously drawn, a field will be displayed for entry anyway.</p>

ENTER VARIABLE USING *vf2* RETURN ESC PGUP PGDN

Gives the operator cursor access to all the displayed variable values in the variable form named *vf2* for purposes of entering one or more values. The operator can signal that all entries have been made by pressing any one of the three keys [ESC], [PgUp], and [PgDn]. The application could interpret [PgUp] to mean "Draw the previous part of a multi-part form next," [PgDn] to mean "Draw the next part of a multi-part form next," and [ESC] to mean "Erase the screen because values have been entered for all parts of a multi-part form." The application can determine which key the operator pressed most recently by interrogating the value of the system variable #RETURN.

ERASE

SYNTAX

ERASE filespec

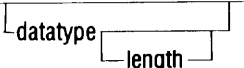
Purpose	The ERASE command deletes the specified file either from the default drive or from the drive you specify. If you do not specify a path, the file is deleted from the current directory or from the root directory of the drive you specify.
Option	You can enter DEL or ERA instead of ERASE.
Comments	<p>Whenever the R> prompt is displayed, you can use the ERASE command. You can use the wildcard characters ? and *. The ? character can be used in place of any one character in a file name. The * character can be used in place of a single character and the characters that follow.</p> <p>This command is similar to the DOS ERASE command.</p> <p>To erase a subdirectory name, use the RMDIR (remove directory) command.</p> <p>See your DOS manual for information on messages.</p>
Examples	<p>ERASE \oldsales*. *</p> <p>Executing this command deletes all files in the directory <i>oldsales</i>.</p> <p>DEL E:\oldfiles\olddata</p> <p>Executing this command deletes the file named <i>olddata</i> from drive <i>e:</i> in the directory <i>oldfiles</i>.</p>

EXIT**SYNTAX****EXIT**

Purpose	Use the EXIT command to leave R:base and return to the operating system.
Comments	The EXIT command automatically CLOSES any database open at the time the command is executed.

EXPAND

SYNTAX

EXPAND tblname WITH colname 

datatype is the data type of the new column (DATE, DOLLAR, INTEGER, REAL, TEXT, or TIME)

length is an optional parameter, necessary only if the data type is TEXT with a length other than eight characters

Purpose The EXPAND command enables you to add a column to the end of an existing table.

Options The *length* is only specified for TEXT data types.

Comments When adding a new column to a table, the type must be specified.

When adding a column that exists in the database to a table, the type and length must be omitted.

The EXPAND command cannot assign keys to columns, nor can it transfer the key designation of an existing column. If the added column should be a key, use the BUILD KEY command.

EXPAND creates a temporary table so your database must have room for at least one table (maximum 39 defined).

Examples EXPAND timecard WITH ssn TEXT 11

Adds the 11-character text column *ssn* to the table *timecard*.

EXPAND custlist WITH mailadr

Adds the existing column *mailadr*s to the table *custlist*.

FILL/NOFILL

To automatically fill trailing column values with nulls when you are in the LOAD mode, at the L> prompt type:

```
SYNTAX
```

```
FILL
```

To eliminate automatic null value filling, at the L> prompt type:

```
SYNTAX
```

```
NOFILL
```

Purpose The FILL command directs R:base to enter a null value for columns that have not been assigned a value. 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, the fill command should not be used.

The NOFILL command turns off the FILL command. When the NOFILL command is entered, R:base will not accept data for a row unless all columns have been assigned a value. The R:base default value is NOFILL.

Comments FILL and NOFILL are only used at the L> prompt in LOAD mode or with the LOAD command and a delimited ASCII file. When FILL is used with an ASCII data file, use the following syntax.

```
LOAD tblname FROM filespec;FILL
```

The FILL command only fills in values that are missing at the end of the row. If you want to assign null values to items in the middle of a row, you have to enter a pair of quotation marks (" ") for TEXT columns or -0- for any data type. For additional information, see the LOAD command.

Example L>FILL
L>100,200,300

If you are loading a table that has five columns but you presently only have information for the first three, you can activate the FILL command to enter null values for the last two columns. The above command sequence can be used instead of:

```
L>100,200,300,-0-,-0-
```

FILLIN

SYNTAX

```
FILLIN varname USING "message"
                                AT scrnrow scrncol
```

message is the message text to be displayed

- Purpose** The FILLIN command is used in a command file to prompt the command file operator to enter a value from the keyboard. The value is stored in an R:base variable.
- Options** *AT scrnrow scrncol* is the screen location to display the first character of the message text.
- Comments** The FILLIN command can also be used to get the operator's choice from a menu that is displayed by the DISPLAY command (see example below).
- The variable is set to the data type that matches the operator entry. If you enter *123*, for example, the variable is typed as INTEGER. If you want the data type as TEXT in this case, you must enclose the numeric response with quotation marks. Entering "*123*" would set the data type to TEXT.
- Examples** FILLIN v2 USING "Enter the name of the database you want to open: "
- Prompts the operator for a database name and stores the response in variable *v2*. The database name can then be used in an OPEN command:
- ```
OPEN .v2
```
- DISPLAY mainmenu ; FILLIN v1 USING "Enter a letter (A-F)" AT 16,20
- Displays the menu text, which is either a block named *mainmenu* in the last procedure file to be used by the command file or in a file named *mainmenu* in the current directory on the default drive. Prompts the operator to enter a menu response in the form of a letter (A through F) at screen row 16, starting at column 20 and waits for a response. Stores the response in R:base variable *v1* and resumes command file execution.

# FORMS

## SYNTAX

FORMS

- Purpose** The FORMS command starts the process by which you can create or modify a table form or a variable form. Enter a 1 to 8 character form name. If the *formname* is not entered at the command line, FORMS prompts for the table name.
- Options** You can use the FORMS command to make either a table form or a variable form. To create a table form, enter a table name when you are prompted to do so. To create a variable form, press the [ENTER] key when you are prompted to enter a table name.
- Comments** An R:base table form is a single page data entry form that is easy to design. You can position headings and explanatory notes anywhere on the screen. These are called the form mask. You also locate the areas where the entered data is displayed. Each table form relates to a specific table in the database. The table must exist before you can define a form for it. Not all columns from the table have to be included on the form. If you only use some of the columns, the columns that are not listed are assigned null values when the form is used for data entry. The definition process for column and variable forms is virtually the same.
- An R:base variable form is an application of the basic table form and is used in command files. Like the table form, you can create a mask anywhere on the screen and locate the areas where the data is to be entered and displayed. Unlike table forms, your application can package and display several single-page forms as if they were one multiple-page form. You can control the prompts that operators will see, and set up different kinds of data input or edit procedures for different levels of users.
- The primary difference between table forms and variable forms is that variable forms must have some type of command file or application generating the variable information and structure to be used by the form. Table forms can stand alone and, since they are associated with a specific table, data input through this type of form is applied only to that table.

## GOTO

### SYNTAX

GOTO *labelname*

*labelname* is the name of a line in the command file to which you want control passed

- Purpose** To cause the next command to be executed in a command file to be the command which directly follows the LABEL *labelname*.
- Comments** The corresponding LABEL command may precede or follow the GOTO command in the same command file. GOTO jumps must be within the same command file.
- Use of the GOTO command should be limited as they execute more slowly than other R:base control structures. The WHILE...THEN...ENDWHILE structure and the IF...THEN...ELSE... ENDIF structure can be used to build most of the command file logic necessary.
- Examples**
- ```
SET VAR v1 TO -1
IF v1 > .v2 THEN
  GOTO exit
ENDIF
LABEL prepare
.
.
.
LABEL exit
QUIT TO caller
```
- When executed, the *GOTO exit* command causes the commands between the label *prepare* and the label *exit* to be skipped and the *QUIT TO* command to be executed. Note that the only way the commands between the labels *prepare* and *exit* will be executed is if the value of *v1* is greater than the value of *v2* or there is a
- GOTO prepare
- statement in the command file.

HELP

SYNTAX

HELP

commandname

commandname is the R:base command for which help text is to be displayed

- Purpose** The **HELP** command enters **HELP** mode so you can browse through the on-line assistance screen displays. The **HELP** *commandname* statement displays the on-line assistance screen for a particular command.
- Options** You do not need to preface the *commandname* with the word **HELP** when you are in the **HELP** mode (**H>**). For example, if you want help for the **SELECT** command when you have the **H>** prompt, type:
- SELECT**
- If you want help for the **SELECT** command when you have the **R>** prompt, type:
- HELP SELECT**
- You may press the **[F10]** key at the **R>** prompt to obtain on-line help.
- Comments** The file that contains all the R:base Help information is named **HELP.RBS**. It must reside on the default directory/drive or on the current DOS path.
- Typing **HELP COMMANDS** will display a screen with all R:base 5000 command names.
- Examples** **HELP ASSIGN**
- Displays the help screen for the **ASSIGN** command.

IF...THEN...ELSE...ENDIF

SYNTAX

```

IF condlist THEN
    then-block
ELSE                               (optional)
    else-block                     (optional)
ENDIF

```

condlist is a set of simple conditions which combine to form a logical statement which is either true or false. Up to ten simple conditions, combined with AND and OR, can be in the logical statement.

then-block is one or more R:base commands which are to be executed when the *condition* is true.

Purpose The IF command is used in a command file to cause a block of commands to be executed only under specified conditions.

Options The ELSE reserved word and the *else-block* are optional. If used, the *else-block* contains one or more commands to be executed when the conditions specified in *condlist* are false.

The IF...ENDIF structure may be on a single line in the command file. For example,

```
IF x = 0 THEN ; SET VAR y to y + 1 ; ENDIF
```

The only exception to this is if the last command in the *then-block* is QUIT. If so, you must place the ENDIF on a separate line.

Comments The IF and THEN must be on the same command line. The IF, ELSE, and ENDIF must be on different command lines.

When the conditions are true, R:base executes all commands between the THEN and ELSE or between THEN and ENDIF, if the optional ELSE is not used.

When the conditions are false and ELSE is used, R:base executes the block of commands between the ELSE and the ENDIF. If ELSE is not used and the conditions are false, no commands between THEN and ENDIF are executed and execution resumes with the command line immediately after the ENDIF.

IF commands can be nested up to ten levels.

Condlist is a set of up to ten simple conditions. There are five kinds of simple conditions as shown in table 4.

Table 4 Simple Conditions

Kind of Simple Condition	Condition is True if ...
varname EXISTS	The value of the variable is other than NULL.
varname FAILS	The value of the variable is NULL.
varname CONTAINS string*	The variable is of data type TEXT, and <i>string</i> is contained as a sub-string within the text string currently stored in the variable.
varname <i>op</i> value <i>op</i> is EQ, NE, GT, GE, LT, LE, =, >, <, <=, >=, <>	The value of the variable has the specified relationship (equal, not equal, etc.) to the value in the condition. (Wild cards may be used in values for TEXT, TIME, and DATE data types.)
varname1 <i>op</i> .varname2 <i>op</i> is EQ, NE, GT, GE, LT, LE, =, <>, >, >=, <, or <=	The values of the two variables have the specified relationship (equal, not equal, etc.).

*You can use *NE *string** to form an effective NOT CONTAINS operator where * indicates wildcard characters and *string* is the text value. CONTAINS is equivalent to *EQ *string**.

Up to ten simple conditions may be combined in a single IF command by using AND and OR to form a logical expression. An example of three simple conditions combined with AND and OR operators into a *condlist* is shown below.

```
IF firstn EQ Mary OR firstn EQ John AND lname EQ Smith THEN
```

```

.
.
.

```

```
ENDIF
```

In the example, the *condition* will be true only when the value of the variable *firstn* is Mary or John and the value of the variable *lname* is Smith.

If you use both AND and OR in a *condition*, you may get confusing results unless you are careful. Parentheses cannot be used in a *condition* and the simple conditions are evaluated from left to right. The above example is equivalent to ((firstn = Mary OR firstn = John) AND lname = Smith).

IF...ENDIF structures may be nested ten deep. An example of three nested IF...ENDIF structures is shown below.

```
IF qtyord GT .lastqty THEN
  IF qtyord NE 0 THEN
    IF change GT 0 AND backord EXISTS THEN
      .
      .
      .
    ELSE
      .
      .
      .
    ENDIF
  ENDIF
ENDIF
```

Examples See above and chapter 15 in the *R:base 5000 User's Manual*.

INPUT

SYNTAX

INPUT filespec

Purpose	The INPUT command changes the input source from your keyboard to a file.
Comments	<p>The default input source of R:base commands and data is the keyboard.</p> <p>The file named in an INPUT command is a command file that will start running when the INPUT command executes. The preferred way to run a command file is to execute a RUN command.</p> <p>To reset the input source to the keyboard, the word QUIT or INPUT KEYBOARD should be included in the input source file.</p> <p>A common use of INPUT is in conjunction with the OUTPUT and UNLOAD commands. After the data and/or structure are UNLOADed to a file, the INPUT command is used to bring the file, which contains LOAD commands, and the data and schema, back into a database.</p>
Examples	<p>INPUT b:my.cmd</p> <p>Causes a file named <i>my.cmd</i> on drive <i>b:</i> to begin processing. This file becomes the source of input to the system.</p>

INTERSECT

SYNTAX

```
INTERSECT tblname1 WITH tblname2 FORMING tblname3         USING collist        
```

tblname1 is an existing table

tblname2 is an existing table

tblname3 is a table to be formed from *tblname1* and *tblname2*

collist is one or more columns from *tblname1* or *tblname2*

- Purpose** The INTERSECT command combines two tables (*tblname1* and *tblname2*) that have one or more common columns into a new table (*tblname3*) that contains only rows in which the values in the common columns are identical.
- Options** *USING...*: This clause lets you specify which columns are to be included in the new table. The clause must include at least one common column. If you do not include the clause, R:base uses the combination of all columns from both tables.
- Comments** If a row from one table has more than one column in common with a row from the other table, every corresponding value in those columns must match or the rows will not be included in the new table.

Examples

INTERSECT emtable WITH pay84 FORMING newtable

Figure 2 shows the results of this example.

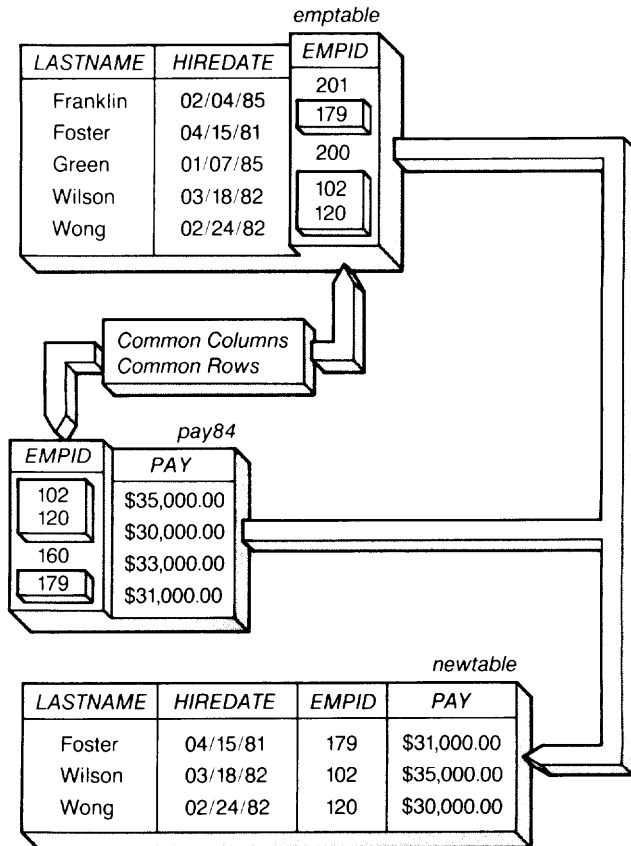


Figure 2 An Example of the INTERSECT command

JOIN

SYNTAX

```
JOIN tblname1 USING colname1 WITH tblname2 USING colname2
... FORMING tblname3
WHERE logicalop
```

tblname1 is an existing table.

colname1 is a column in *tblname1*.

tblname2 is an existing table.

colname2 is a column in *tblname2*.

tblname3 is a table to be created from *tblname1* and *tblname2*.

logicalop is a specified comparison of the values from *colname1* to values from *colname2*. *Logicalop* is one of the following:

Table 5 Logical Operations

Logical Operator	Rows are included if...
EQ	colname1 equals colname2 (default)
NE	colname1 is not equal to colname2
GT	colname1 is greater than colname2
GE	colname1 is greater than or equal to colname2
LT	colname1 is less than or equal to colname2
LE	colname1 is less than or equal to colname2

Purpose	The JOIN command forms a new table (<i>tblname3</i>) by combining rows from two existing tables (<i>tblname1</i> and <i>tblname2</i>). The rows in the new table are based upon the comparison of a column from each of the existing tables. You must specify the columns to be compared. Rows satisfying the comparison condition are combined to form the new table.
Comments	When R:base finds a match between the two specified columns, it creates in the new table a row that is a combination of all columns from the row in <i>tblname1</i> and its corresponding row in <i>tblname2</i> . The names of the columns you specify in the command should be different. The data type and length, however, must be the same.

For setting up the comparison, you use a unique form of the WHERE clause. The purpose of the clause is to state the condition you want used for the comparison of the columns. If you choose a condition of *equals*, you may omit the WHERE clause, since this is the default. To specify any other condition, use the WHERE clause.

Example

JOIN repsales USING sales WITH contest USING amount FORMING + winners WHERE GT

Figure 3 shows the results of this example.

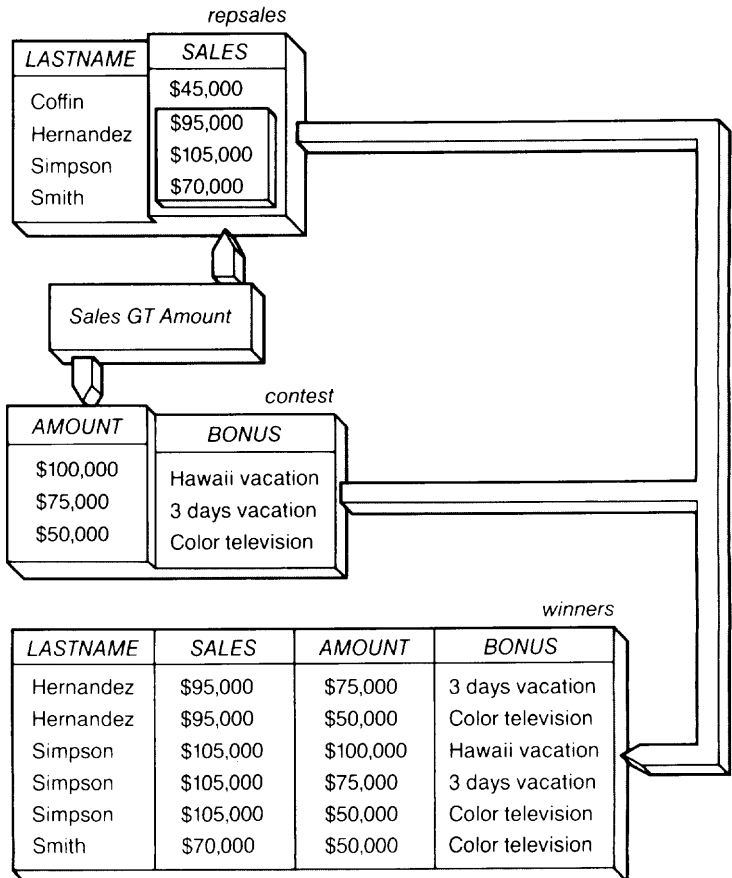


Figure 3 An Example of the JOIN command

LABEL

SYNTAX

LABEL *labelname*

labelname is a one to eight character string that can be used in a GOTO statement to designate a specific LABEL.

Purpose LABEL *labelname* is used to identify a command line to which control may be passed from a GOTO statement in a command file.

Examples IF *ctrvar* = 999 THEN
 GOTO *endproc*
ELSE
 <process>
ENDIF

·
·
LABEL *endproc*
 <process>

On the basis of the IF...THEN statement, if the variable *ctrvar* is equal to 999 then control is passed to the processing steps following the LABEL command that defines the label *endproc*. If *ctrvar* does not equal 999 then the processing steps following the ELSE statement are performed.

LIST

SYNTAX

LIST listtype

listtype is one of the following: ALL, COLUMN NAMES, DATABASES, FORMS, REPORTS, RULES, TABLES, *tblname*

Purpose	The LIST command displays or prints basic information about the database or available databases.
Options	<p>The information displayed or printed depends on the keyword entered with LIST. The keywords, their short form, and their purpose are:</p> <p>ALL—Lists all the tables, password status, row counts, column names, lengths, data types, and keys in the database</p> <p>COLUMN NAMES—Lists all column definitions in the open database</p> <p>DATABASES—Lists all databases on the default directory</p> <p>FORMS—Lists all form names defined for the open database</p> <p>REPORTS—Lists all report names defined for the open database</p> <p>RULES—Lists the rules defined for the open database and the status of the rule checking mode</p> <p>TABLES—Lists all table names in the open database</p> <p><i>tblname</i> Lists information about the specified table</p>
Comments	<p>If the LIST command is entered by itself, the LIST TABLES command is executed.</p> <p>The column names are always sorted alphabetically when the LIST COL version is used. Tables are listed in the order they were created on the database.</p> <p>Column numbers shown with the list command can be entered in commands instead of column names. Enter #<i>c</i> where <i>c</i> is the column number.</p> <p>If a table is password protected, you must enter the appropriate password before entering the LIST command.</p>

LOAD

SYNTAX

LOAD *tblname*

WITH PROMPTS

FROM *filespec*

FROM *filespec* AS ASCII

USING *collist*

Purpose

The LOAD command adds data to a table within the open database.

Options

In the simplest form (LOAD *tblname*), R:base prompts with L>. Enter the column values in the order the columns are defined in the table. Separate columns with blanks or commas, and enclose text entries that contain blank spaces with quotation marks.

WITH PROMPTS: Loads data into the specified table from keyboard entries. R:base prompts for each column by displaying the column name and its data type.

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

FROM *filespec* AS ASCII: Loads data into the specified table with data from an external ASCII delimited file. The ASCII delimiter format must be one of the following forms:

"val 1",val2,"val3"...: The quotation marks optional for all non-TEXT datatypes and for TEXT entries that do not contain blank spaces

"val 1" val2 "val3"...: Spaces can be substituted for commas as the delimiter character.

USING... is an optional clause to designate specific columns and the loading order rather than loading all columns in the table.

The FILL and NOFILL commands can be used in LOAD mode. See FILL/NOFILL in this Reference for information.

The CHECK and NOCHECK commands can be used in LOAD mode. See CHECK/NOCHECK in this Reference for information.

Comments

The LOAD *tblname* command adds data to a table, row by row, without using a data entry form and without being prompted for each data item. When data loading is complete, END exits LOAD mode.

If you specify the WITH PROMPTS keywords, you are prompted for column values with the column names and types. End by pressing [ESC].

For any kind of LOAD you can identify the specific columns to be loaded with a USING *collist* clause.

For any kind of LOAD except LOAD WITH PROMPTS, you can specify a *filespec*.

For additional information on converting files to R:base, see chapter 13.

Examples

LOAD custlist FROM cust.dat AS ASCII

This command loads data into table *custlist* from the external ASCII delimited file *cust.dat*.

LOAD anytable WITH PROMPTS USING colid colname coladdr colcity + colstate colzip

This command loads data into columns *colid*, *colname*, *coladdr*, *colcity*, *colstate*, and *colzip* in table *anytable*. R:base prompts for each column with the column name and type.

MKDIR

SYNTAX

MKDIR d: path directory

d: is a drive designation

path is a series of one or more directory names that lead from the root directory to a specified directory

Purpose	The MKDIR command creates a new subdirectory on the drive you specify or, if you do not specify the drive, on the current drive.
Options	You can use the short form MD for MKDIR.
Comments	<p>You may make as many subdirectories as you wish. This is indicated in the syntax by the word <i>path</i>. Note, however, that DOS limits the length of a path from the root to a maximum of 63 characters, including backslashes.</p> <p>This command is similar to the DOS MKDIR command. See your DOS manual for information on messages.</p>
Examples	<p>MKDIR e:\newdata</p> <p>Executing this command creates a directory, <i>newdata</i>, on drive <i>e:</i>.</p> <p>MD \sales\august</p> <p>Executing this command creates a new subdirectory, <i>august</i>, in the directory called <i>sales</i>. Since the directory and subdirectory are on the default drive, the drive is not specified.</p>

MOVE**SYNTAX**

```
MOVE nchar FROM varname1 AT chrpos1 TO varname2 AT chrpos2
```

nchar is an integer between 1 and 1500 which specifies the length of the substring to move

varname1 is the text variable from which to move the substring (the source string)

chrpos1 is an integer between 1 and 1500 which specifies the character position within the source string from which to start moving the substring

varname2 is the TEXT variable to which to move the substring (the target string)

chrpos2 is an integer between 1 and 1500 which specifies the character position within the target string to which the first character of the substring is moved

Purpose To move a substring from one TEXT variable to another.

Comments The first character in both the source and target strings is character position 1.

Both the source string and the target string are assumed to be 1500 characters long, blank-filled to the right.

If (*chrpos2* + *nchar*) is greater than 1500, then those characters in the source string which will fit into the target string are copied into the target string. The excess source string will not appear in the target string.

If (*chrpos1* + *nchar*) is greater than 1500, then only those characters from source string character position *chrpos1* to the end of the source string are copied into the target string.

Examples

```
MOVE 5 FROM savetext AT 11 TO zipcode AT 1
```

If variable *savetext* held "aaaaabbbb96002ccccc", this command would move the five characters beginning in the eleventh position (96002) into variable *zipcode* beginning at the first position. The variables *savetext* and *zipcode* must be TEXT type variables.

NEWPAGE

SYNTAX

NEWPAGE

- | | |
|----------|---|
| Purpose | The NEWPAGE command sends a form-feed character to your current output device. |
| Comments | If your output device is the computer screen, typing NEWPAGE clears the screen and moves the R> prompt and cursor to the top of the screen. If your current output device is a printer, NEWPAGE causes the paper to advance one page. |

NEXT

SYNTAX

```

NEXT #n
      |
      |____varname_____
  
```

#n is a route number (1, 2, or 3) which can be set by the SET POINTER command

Purpose	The NEXT command advances the R:base row pointer to the next row in the table associated with the route number as defined by a SET POINTER command.
Options	<i>varname</i> can be used to detect when there are no more rows in the route. Each time the NEXT command finds another row, it sets <i>varname</i> to 0. When there are no more rows, it sets <i>varname</i> to -1. Note that this can happen because the NEXT command has moved the pointer past the last row in the table route or there were no rows to begin with.
Comments	The route number identifies a set of rows that are to be processed. The route number and associated table that are defined with the SET POINTER command can be the entire table (SET POINTER without a WHERE clause) or a portion of the table (SET POINTER with a WHERE clause). In addition, the order of the rows is determined by the SORTED BY clause used with the SET POINTER command. If no SORTED BY clause is included, the order is the natural order of the rows in the table. See the SET POINTER command description in this Reference and chapter 15 of the <i>R:base 5000 User's Manual</i> .

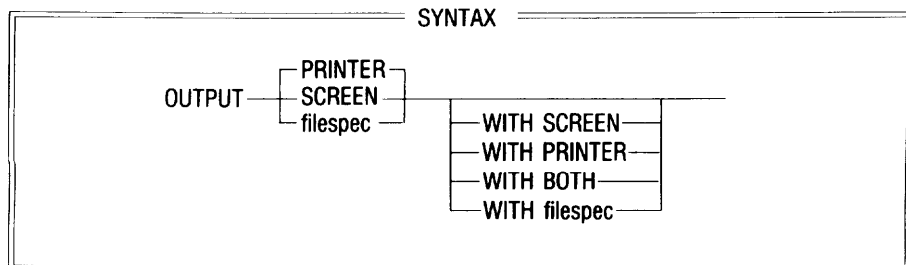
OPEN

SYNTAX

OPEN dbspec

- Purpose** The OPEN command opens a previously defined database for use. You must open a database in order to query, add, edit, or delete data or to print reports.
- Comments** If the database is not in the default directory, add the drive and path before you type the database name. Only one database may be open at a time.
- An open database is closed when:
- Another database is opened
 - The CLOSE command is executed
 - The EXIT command is executed
- If you want to set up a new database or modify the structure of an existing database, use the EXPRESS or the DEFINE command.
- Example** OPEN c:\tools\wp\letters
- If the database *compuco* is open and you type this command, *compuco* is closed and *letters*, located on drive *c:* in the \tools \wp\ subdirectory, is opened.

OUTPUT



- Purpose** The OUTPUT command directs messages and the results of commands to a file, printer, screen, or any combination of the three. The primary output device can be a file, the PRINTER, or the SCREEN. The secondary device, if any, can be the PRINTER, the SCREEN, a file, or BOTH (referring to PRINTER and SCREEN). Messages and output continue to go to the devices you have specified until you enter another OUTPUT command.
- Options** *PRINTER, SCREEN, or filespec*: Indicates the primary output device.
WITH PRINTER, SCREEN, BOTH or filespec: Indicates the secondary output device. WITH BOTH refers to PRINTER and SCREEN.
- Comments** Normally, you use the OUTPUT command to send a very specific set of information to the output device. You must remember to reset the output device to SCREEN after the processing is complete. The default output device is the screen when you first enter R:base.
- The word TERMINAL may be used instead of SCREEN.
- If the output device is SCREEN and the data requires more than one screen to be displayed, R:base pauses at each screen.
- If the primary output device is a file and the WITH SCREEN option is used, R:base pauses at the end of each screen.
- You cannot combine two *filespecs* as in:
- OUTPUT filespec WITH filespec

If you want to send your commands as well as the result of the commands to an output device, type

SET ECHO ON

before beginning the OUTPUT session. When the session is over, type

SET ECHO OFF

Examples

OUTPUT a:name.dat WITH PRINTER

Outputs subsequent data and messages to file *name.dat* residing on drive *a:* and simultaneously prints the file contents on the printer.

OUTPUT PRINTER WITH SCREEN

Outputs subsequent data and messages to the printer and simultaneously displays the data and messages on the screen with pauses between screens, if needed.

OWNER

SYNTAX

OWNER password

password is the password you have chosen for the database

Purpose The OWNER command is used within DEFINE mode (D>) to specify the owner password for a database.

Comments If you specify an owner password, operators must enter the password to modify the database structure with the DEFINE, RELOAD, and UNLOAD commands. Operators without the password are only able to view or modify the data in the database.

If an explicit owner password is not specified by executing an OWNER command in the DEFINE module, R:base displays the word NONE when the database rules are shown on the screen.

Anyone who knows the owner password can change it by using the RENAME OWNER command at the R> prompt or by using the OWNER command in D> mode.

Keep a record of the owner password in a safe place away from your computer. If you lose the owner password, it is impossible to search the database structure to determine the password. You also cannot change the database structure.

An OWNER password should first be defined if you want to define read and modify passwords (RPW and MPW) for tables in the database. Since the OWNER password has precedence over the RPW and MPW passwords, if it is not defined then the read and modify passwords will have no effect.

Passwords can be from 1 to 8 characters, with no blank spaces. The first character of the password cannot be a number or special character.

PACK

SYNTAX

```
PACK           
      |_____|
      dbspec
```

- Purpose** When the **REMOVE** or **DELETE** commands are used to remove tables or delete rows, the information is no longer stored in the database. The disk space that was occupied by the deleted information, however, is unusable because the R:base commands do not recover the space. The **PACK** command recovers the unusable space from the database to the disk.
- Options** If no database name is defined, R:base packs the currently open database.
- Comments** Include the drive designation and path before the database name if the database is not on your current directory.
- The end result of **PACK** command execution is the same as executing a **RELOAD** command. However, **PACK** should be used by owners of very large databases and users with little disk space because it does not duplicate the database on the disk during execution.
- If the **PACK** command is interrupted, either because of human error or mechanical malfunction, the database can be destroyed, therefore be sure to back up your database before you use the **PACK** command.

PASSWORDS

SYNTAX

PASSWORDS

- Purpose** The PASSWORDS command starts the process of assigning read and write passwords to tables.
- Options** To define a password, at the D> prompt use the following syntax:
OWNER password
- Comments** The owner password must exist before you define modify and read passwords (MPW and RPW). To enter an MPW or RPW, at the D> prompt type:
PASSWORDS
- R:base responds with a D> prompt. Choose the type of password you wish to define (MPW or RPW) and use the following syntax:

SYNTAX

```

RPW  }
      } FOR tblname IS password
MPW  }

```

The OWNER password controls access to the database structure. If an OWNER password is defined, only persons with that password can use the CHANGE COLUMN, DEFINE, RELOAD, RENAME, or UNLOAD commands.

The OWNER password allows you to read or modify, including modifying the database structure. The MPW password allows you to read or modify tables. The RPW password allows you to read only. You can define one MPW and one RPW for each table. If you to define an RPW, it will only be effective if you define an MPW.

Examples OWNER ppjrwpp

This defines *ppjrwpp* as the owner password for the open database.

MPW FOR transx IS zzjackzz

This defines *zzjackzz* as a modify password for the *transx* table.

PAUSE

SYNTAX

PAUSE

Purpose	The PAUSE command is used to suspend execution of a command file until the operator responds by pressing a key on the keyboard.
Comments	Before the PAUSE command is executed, a TYPE or WRITE command should be used to send a message directing the user to press a key to continue. Execution will continue as soon as any key is pressed.
Example	<pre>WRITE "Illegal entry, try again. Press any key to continue." PAUSE</pre> <p>This displays a message to the user then waits until a key is pressed to continue execution.</p>

PRINT

SYNTAX

```
PRINT rptname [SORTED BY collist] [WHERE condlist]
```

rptname is the valid name of a generated report stored in the REPORTS table

- Purpose** The PRINT command is used to output a report created by the REPORT processor.
- Options** *SORTED BY...*: The report detail lines may be sorted by up to ten columns. If breakpoints are indicated in the report, the rows are automatically sorted by the breakpoint columns. If a SORTED BY clause is used in the PRINT command, then the breakpoint sorts are not used.
- WHERE...*: Allows you to limit the rows printed on the report.
- Comments** If the report is to be sent to a file or the printer, precede the PRINT command with an OUTPUT command designating the output device—a file or the printer. The default output device is the terminal screen.
- If the report has breakpoints (subheadings or subfootings), then R:base automatically sorts by the defined breakpoint columns. Overriding the breakpoint sort order may cause unpredictable results.
- Examples** PRINT sales SORTED BY transid
- Prints a sales report sorted by transaction number order to the screen.
- OUTPUT PRINTER**
- PRINT sales SORTED BY custid prodid WHERE price GE \$1000.00
- Prints a sales report sorted by customer with a subsort by product where the product price is greater than or equal to \$1000.00. The output is directed to the printer by the OUTPUT command. When the report is finished printing, you should redirect output to the screen with the OUTPUT SCREEN command.

PROJECT

SYNTAX

```
PROJECT tblname2 FROM tblname1 USING 

|         |
|---------|
| collist |
| ALL     |

 ...  
... 

|                   |
|-------------------|
| SORTED BY collist |
|-------------------|



|                |
|----------------|
| WHERE condlist |
|----------------|


```

tblname1 is an existing table

tblname2 is a new table to be formed from *tblname1*

- Purpose** The **PROJECT** command creates a new table (*tblname2*) from an existing table (*tblname1*). The new table can be either an identical copy of the original table or a subset of the original table.
- Options** ***SORTED BY...***: This clause is used to sort the rows by the specified columns.
- WHERE...***: This clause limits the rows selected.
- Comments** You must include the **USING** clause. If you want a subset of columns from the existing file, name them. If you want all columns, use the word **ALL** in the clause.

Example PROJECT newtable from emptable USING empid lastname SORTED BY +
empid WHERE hiredate LT 01/01/85

Figure 4 shows the results of this example.

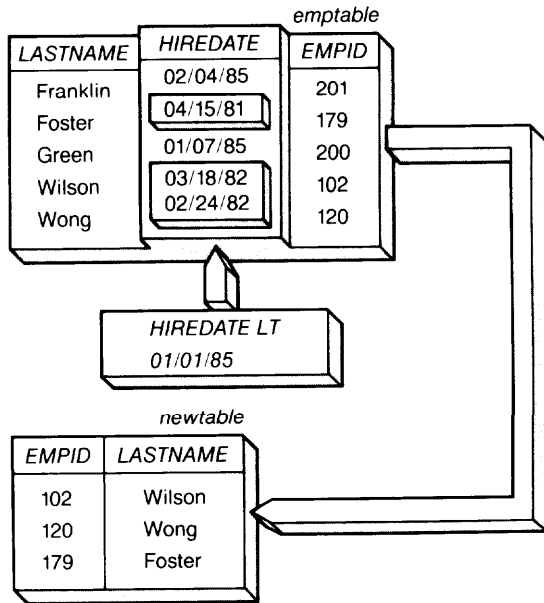


Figure 4 An Example of the PROJECT Command

PROMPT

SYNTAX

PROMPT

commandname refers to the R:base command you want to execute with the assistance of the PROMPT mode

- Purpose** The PROMPT command provides on-line assistance for R:base commands. R:base assists you by prompting you for the command parameters and then creating the correct command syntax based on those parameters.
- Options** Enter a command name after the word PROMPT on the command line for assistance in executing one command.
- Comments** When you enter the PROMPT mode from the menu, R:base displays a list of commands that have on-line prompts available.
- When you enter the PROMPT mode from the command mode, R:base displays a summary of the command and prompts for the information needed.
- The amount of space left after the prompt varies depending upon the type of information to be entered but never exceeds 73 characters. If you need to enter more than 73 characters (for example, in a complex WHERE clause), you must enter the commands directly instead of using the PROMPT mode.
- The file that contains all the R:base prompts is named PROMPT.RBS. It must reside on the default drive or in the directory referenced by the DOS PATH command. If you want to create your own prompt screens, see chapter 15 of the *R:base 5000 User's Manual*.

Not all R:base commands have prompts. In general, only the commands that are executed interactively from the keyboard have prompts. You can use prompts to execute the following R:base commands.

APPEND	COPY	EXIT	NEWPAGE	RELOAD	SHOW
ASSIGN	DATE	EXPAND	OPEN	REMOVE	SUBTRACT
BUILD	DEFINE	FORMS	OUTPUT	RENAME	TABLES
CHANGE	DELETE	INPUT	OWNER	REPORTS	TALLY
CHDIR	DIR	INTERSECT	PACK	RMDIR	TYPE
CHKDSK	DISPLAY	JOIN	PASSWORDS	RULES	UNION
CLOSE	EDIT	LIST	PRINT	RUN	UNLOAD
COLUMNS	ENTER	LOAD	PROJECT	SELECT	USER
COMPUTE	ERASE	MKDIR	RBEDIT	SET	

Type **PROMPT** to display this list on the screen. The screen also provides a line with your options.

Example

PROMPT SELECT

Displays the prompt screen for the **SELECT** command. After you are satisfied that you have responded to the prompts in a way that describes the task you want the command to perform, press [G] (for Go) and the **SELECT** command is executed.

QUIT

SYNTAX

QUIT

TO filespec

Purpose	The QUIT command closes all currently open command files and closes all IF and WHILE blocks.
Options	<i>TO filespec</i> : Specifies the command file which is to run after the QUIT command executes. When this option is not used, R:base returns to the command mode (R> prompt).
Comments	The stacks maintained by R:base to control the nesting of command files (by the execution of RUN and INPUT commands), and the stacks for nesting IF and WHILE blocks are cleared when the QUIT command executes.
Examples	<p>QUIT</p> <p>Terminates all open command files, IF blocks, and WHILE blocks. Returns the system to command mode (R> on the screen).</p> <p>QUIT TO getval.inv</p> <p>Terminates all open command files, IF blocks, and WHILE blocks and runs the command file named <i>getval.inv</i>.</p>

RBEDIT

SYNTAX

RBEDIT

[*filespec*]

filespec refers to an ASCII text file

- Purpose** Use the RBEDIT command to create or to edit small text and command files within R:base. RBEDIT prompts for a file name if it is not included in the command line. There is also a stand-alone version of RBEDIT which can be executed from DOS.
- Options** When the *filespec* option is not used, RBEDIT prompts for the name of the file to edit.
- When the *filespec* option is used, and the file already exists, the contents of the file will be displayed on the screen for editing. If the file does not exist, a blank screen will be displayed.
- Comments** When you are through editing, press [ESC]. RBEDIT displays a menu which gives you the opportunity to save the new version of the file on disk under its old filename or under a new filename which you enter at that time.
- When RBEDIT is used within R:base, you can edit approximately 130 full lines of text. Used outside of R:base as a stand-alone editor, you can edit approximately 800 full lines of text.
- RBEDIT is a full-screen editor. Table 6 shows all of the keys RBEDIT recognizes and their function.

Table 6 RBEDIT Keys and Their Functions

To browse through pages:

[Home]	Display the first page (first 23 lines) of the text file
[End]	Display the last page of the text file
[PgUp]	Display the previous page
[PgDn]	Display the next page

To move the cursor in text line units:

[↓]	Move down one line
[↑]	Move up one line
[Ctrl][→]	Move to end of current line
[Ctrl][←]	Move to start of current line
[ENTER]	Move to start of next line

To move the cursor in character units:

[→]	Move one character to the right
[←]	Move one character to the left

To edit a line:

[F1]	Insert a line above the current line
[F2]	Delete the current line

To edit a character:

[Ins]	Insert a blank character
[Del]	Delete a character
[F4]	Turn on or turn off character repeat

[Alt] and a ten-key numeric	Allows entry of special characters by ASCII code
-----------------------------	--

Examples See chapter 15 in the *R:base 5000 User's Manual* for an example of RBEDIT usage.

RELOAD

SYNTAX

RELOAD *newdbspec*

newdbspec is a valid database name

Purpose The RELOAD command makes a copy of a database from an existing database but does not copy deleted rows, keys, or tables.

Comments Before using the RELOAD command, use the COPY command to make a duplicate copy of your database.

When rows or keys have been deleted, or tables removed, the disk space that the deleted information occupied is not available until a RELOAD or PACK command is executed.

If you wish to reload a database on the same disk as your original copy, you must assign a different name to the new database when you enter the RELOAD command. If there is not enough room on the disk to store the reloaded database, use the PACK command instead.

If you wish to reload a database on a different disk or directory, you may assign the new copy the same database name. Be sure to specify the new drive or directory when you enter the command.

Examples RELOAD c:\rbase\newbase

This command reloads the currently open database and assigns it the name *newbase* in the *rbase* directory of drive *c*:

REMOVE

SYNTAX

REMOVE tblname

Purpose	The REMOVE command removes the table from the open database and any data (rows) in it become unavailable.
Comments	<p>If a column definition in the deleted table is unique, the definition is removed.</p> <p>After executing the REMOVE command, use the PACK or RELOAD command to restore unusable disk space.</p>
Example	<p>REMOVE custlist</p> <p>Removes the table <i>custlist</i> from the open database.</p>

REMOVE COLUMN

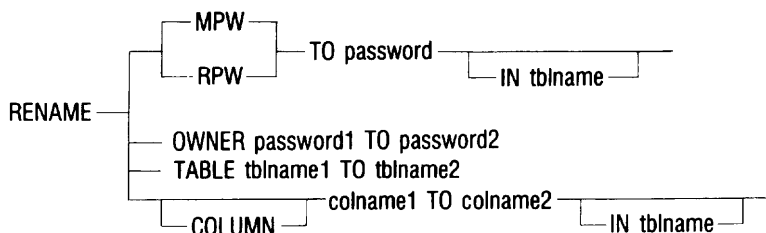
SYNTAX

REMOVE COLUMN colname FROM tblname

- Purpose** The REMOVE COLUMN command deletes a column from an existing table.
- Comments** If the column is only located in one table, the structure and data for the column are deleted from the database structure when the column is removed. If the column is located in more than one table, the column and its data are removed from the specified table only.
- Example** REMOVE COLUMN ext FROM salerep
Removes column *ext* from table *salerep*.

RENAME

SYNTAX



password is the new MPW or RPW password

password1 is the owner password to be changed

password2 is the new owner password

tblname1 is the table name to be changed

tblname2 is the new table name

colname1 is the column name to be changed

colname2 is the new column name

tblname is the table in which to rename *colname1*

Purpose	The RENAME command is used to change the modify, read, or OWNER passwords, and table and column names.
Options	<i>IN tblname</i> : If this clause is not included when renaming a column, all occurrences of the column name in all tables are changed to the new column name. If not included when renaming modify (MPW) and read (RPW), then all table passwords are changed.
Comments	When you rename a column, it can be renamed in an entire database, or only in a single table. If you wish to rename a column throughout a database and the database is protected by an owner password, enter the OWNER password with the USER command before using the RENAME command.
Examples	<p>RENAME transid TO transxno IN transx</p> <p>Renames a column from <i>transid</i> to <i>transxno</i> in the <i>transx</i> table.</p> <p>RENAME OWNER opass TO newpass</p> <p>Renames the owner password from <i>opass</i> to <i>newpass</i>.</p> <p>RENAME MPW TO newpass IN transx</p> <p>Changes the modify password to <i>newpass</i> for table <i>transx</i>.</p>

RENAME (DOS FILE)

SYNTAX

RENAME *filespec1 filespec2*

filespec1 is the current file name and optional drive and directory designation

filespec2 is the new name for the file

- Purpose** This command changes the DOS filename that you specify first (*filespec1*) to the filename that you specify second (*filespec2*).
- Options** You can use the short form REN for RENAME.
You can use the global filename characters ? and *. The ? character can be used in place of any one character in a filename. The * character can be used in place of a single character and the characters that follow.
- Comments** With this command, the file is renamed. It is altered in no other way. It remains in the same directory and on the same drive.
Whenever the R> prompt is displayed, you can use the RENAME command to change the name of any DOS file.
- Examples** RENAME customer customer.dat
Executing this command changes the name of the file *customer* to *customer.dat*.
RENAME e:\salesdep\records olddata
Executing this command changes the name of the file *records* to *olddata*. The file is on drive *e:* in the directory *salesdep*.

REPORTS

SYNTAX

```
REPORTS [ rptname ]
```

rptname is the report name

Purpose	Use REPORTS to generate and edit report formats.
Options	<p>This command prompts for a 1-8 character report name if the report name was not entered on the command line. If the report is new, it also prompts for the table name from which the report data is to be drawn.</p> <p>Once the REPORTS mode is invoked, the following options are provided:</p> <ul style="list-style-type: none">Edit — Edit the reportDefine — Define variablesLocate — Locate columns and variables on the reportMark — Mark heading, detail, footing lines, and breakpointsSet — Set the report page lengthHelp — Display on-line assistance for report generationQuit — Quit and save, discard, or return to report definition. <p>The Edit, Locate, and Mark options are required to produce a report. The rest are optional.</p>
Comments	<p>The REPORTS command is used to enter the REPORTS processor. This is a menu-oriented subsystem with entry screens provided for editing, column and variable placement (location), and report line definition (marking).</p> <p><i>Editing the Report:</i> The E option displays an edit screen which is used to fill in static information on the report such as column headings. A report can be up to 131 columns wide.</p> <p><i>Locating Columns and Variables:</i> The L option displays the Locate Selection Menu which provides options to: 1) locate new columns and variables or 2) relocate, change, or delete existing column and variable locations.</p>

Marking Heading, Detail, and Footing Lines: The Mark option is used to indicate which lines of the report are to appear at the top of each report page (heading), which lines hold data derived from the report's specified table or from defined variables (detail), and which lines are to appear at the end of the report (footing). Marking is done by typing an H, D, or F in the reverse video column displayed at the left edge of the report. R:base displays back HR for report heads, Hn for subheads, FR for report footings, and Fn for subtotals.

Select Mark to display the submenu selections:

- Report — Set report headings and footings.
- Page — Sets page headings and footings.
- Detail — Mark or re-mark detail lines.
- Break — Adds or deletes break columns for each level breakpoint.

Defining Variables: Up to 40 variables may be defined for each report. The D option is used to make calculations such as totals or subtotals which are not held in the table but may be calculated from columns in the specified table.

Printer control options are defined by a variable which is then located on the report wherever you want the printer options to take over. Use the decimal equivalent of the ASCII codes for the printer control options. See your printer manual for the necessary control codes.

Special options are:

REORDER — REORDER varname position

Allows you to change the position of a variable setting.

DELETE — DELETE varname

Allows you to delete a defined variable.

TYPE — TYPE varname vartype

Allows you to change the data type of a defined variable.

Lookup — varname = colname IN tblname WHERE condition

Allows you to assign the value of a column in a different table to a variable.

Setting Page Length: The Set option is used to set the page length of the report and printer control options, if any. The default page length is 66 lines and may be changed to any length up to 999 lines. A setting of 0 indicates no page breaks are to occur. The minimum length unless 0 should be: header lines + footer lines + marked detail lines + marked break subheads and subtotals.

RETURN

SYNTAX

RETURN

Purpose The RETURN command is used in a subordinate command file to return control from the subordinate file to the command file that called it.

Comments The next command executed in the calling command file will be the command immediately following the RUN or INPUT statement that called the subordinate file. (Such subordinate files are usually called subroutines.)

If the command file was RUN or INPUT from the keyboard, when RETURN is executed, it returns to the R> prompt.

Examples Suppose you have a command file named *cmdfil1*:

```
*(CMDFIL1)
WRITE "This command file (cmdfil1) will call cmdfil2"
RUN cmdfil2
WRITE "cmdfil1 is in control"
QUIT
```

Suppose you have a command file named *cmdfil2*:

```
*(CMDFIL2)
WRITE "cmdfil2 has started running"
RETURN
QUIT
```

When you type, at the R> prompt, the command:

```
RUN cmdfil1
```

the following messages appear on the screen:

```
This command file (cmdfil1) will call cmdfil2
cmdfil2 has started running
cmdfile1 is in control
```

RMDIR

SYNTAX

```

RMDIR [d:] path directory

```

d: is the drive designation of the directory to be removed

path is the path of the directory to be removed

directory is the directory to be removed

- Purpose** The RMDIR command removes a directory from the current drive, if you do not specify a drive, or from the drive that you specify.
- Options** You can use the short form RD for RMDIR.
- Comments** Before you can use this command, the directory that you are going to delete must be empty. To empty the directory, use the ERASE command.
- When the path includes one or more directories, the last directory name in the path is the one that will be removed.
- You cannot use this command to remove the root directory or the current directory.
- This command is similar to the DOS RMDIR command. See your DOS manual for information on messages.
- Examples** RMDIR \salesdep\ytdsales\olddata
- Executing this command removes the directory entry named *olddata* from the directory path *salesdep\ytdsales*.
- RD e:\lastyear
- Executing this command removes the directory named *lastyear*. The directory is located on drive *e:*.

RULES

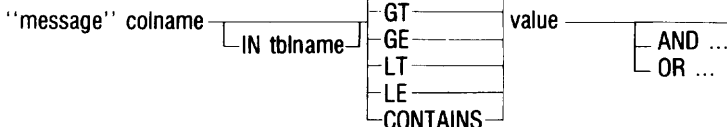
SYNTAX

RULES

To define rules by comparison of a column to a constant value:

SYNTAX

RULES

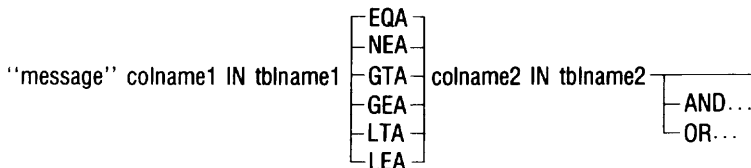


"message" is an error message entered in quotes

tblname is the table in which the column is defined (omit to allow the rule to apply to all tables in which the column is defined)

To define rules by comparison of column values:

SYNTAX



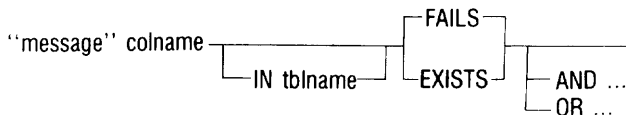
"message" is an error message entered in quotes

colname1 is the column whose entry rules are being defined

tblname1 is the table in which the column is defined

tblname2 is the table in which the comparison column is defined

SYNTAX



- Purpose** Use the RULES command and a set of rule specifications to define data entry and modification validation rules for the columns in your database. You can only enter rules when you are within the DEFINE mode (D>).
- Comments** Up to ten additional comparisons can be made using the operators AND and OR. All comparisons must apply to the first table defined with the RULES command.
- Data validation rules are defined in the DEFINE mode.
- Each rule has its own unique definition and corresponding error.
- If you defined an owner password for the database, you must enter the password before you are allowed to add rules.
- See the SET command for information on turning rule checking on and off.
- Examples** "Not a West-Coast State" state EQ "CA" OR state EQ "OR" OR state EQ + "WA"
- This rule checks the *state* column in all tables and requires that the state be CA, OR, or WA. If *state* is not one of the three choices, the error message is displayed.
- "duplicate employee number" empid IN empfile NEA empid IN empfile
- This rule prevents the same employee number from being entered more than once. An error message is displayed if the employee number (*empid*) entered equals an existing entry.

RUN

SYNTAX

```
RUN cmdfile [IN procfile] [USING paramlist]
```

cmdfile refers to the name of an existing command file or a command block in a procedure file. This may be a filename that refers to an ASCII command file or it may be the name of a binary version of a command file compiled during RCOMPILE.

Purpose Use the RUN command to execute command blocks, command files, or macros requiring passed parameters.

Options The simplest form of the command is equivalent to the command:

INPUT *cmdfile*

Both the simplest form of the RUN command and the INPUT command can be used to run an ASCII command file that was stored by an editor such as RBEDIT.

However, only the RUN command can be used to run a binary command file or procedure file created by RCOMPILE, and only the RUN command has parameters.

IN procfile: This clause names the procedure file in which the binary form of the command block is stored.

USING paramlist: This clause lists the values which will be used by the command file when it executes (see Example). There may be up to nine values in the parameter list. The first value in the parameter list is referenced in the macro file as %1, the second value as %2, and so on. They are treated just like other variables. To reference the contents of these variables, preface the variable name with a dot (.), for example .%1.

Comments A procedure file is a compiled binary file that contains stored menu, screen, and command blocks. See chapter 16 if you need more information about RCOMPILE and the procedure files it generates.

Examples

RUN mycmdfil

If there is an active procedure file, RUN attempts to locate a command block named *mycmdfil* and execute it. If no such block name can be found in the active procedure file, or there is no active procedure file, RUN locates a file named *mycmdfil* in the current directory of the default drive and executes it.

RUN mycmdfil IN myprocfl

Executes the command block named *mycmdfil* in the procedure file named *myprocfl*.

RUN mycmdfil IN myprocfl USING 10

Executes the command block named *mycmdfil* in the procedure file named *myprocfl*, supplying the parameter value 10.

For example, suppose the following R:base commands, which use the R:base system variable #TIME, made up a timing procedure called *mycmdfil*:

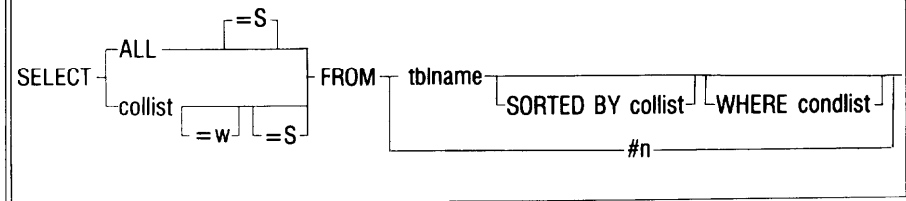
```
SET VARIABLE start TIME
SET VARIABLE wait INTEGER
SET VARIABLE wait TO 0
SET VARIABLE start TO .#TIME
WHILE wait LT .%1 THEN
    SET VARIABLE wait TO .#TIME - .start
ENDWHILE
QUIT
```

If *mycmdfile* had been compiled and placed in a procedure file called *myprocfl* then the RUN command shown in the example above would cause *mycmdfil* to run for ten seconds. The following RUN command would cause *mycmdfil* to run for a five second time delay:

RUN mycmdfil IN myprocfl USING 5

SELECT

SYNTAX



- Purpose** The SELECT command retrieves and displays rows of data from a table.
- Options** Each column name may have =S or =w appended. =w is an integer that specifies the column display width. =S indicates summation of the column or columns.
- FROM #n:* Specifies the starting row indicated by a route but selects all rows following, not just the route rows.
- SORTED BY...:* This clause specifies the order in which to display the retrieved rows.
- WHERE...:* A clause that specifies which rows of data within the table to display.
- Instead of entering a column name, you can enter #c where c is the column number shown when columns are listed.
- Comments** Specify the table that contains the columns you want to view.
- To view all columns, use ALL for the column name.
- To sum a column, append =S to the column name. To set the width of a column, append =w to the column name, where w is a number from 1 to 132. The maximum column display width is 132 characters.
- If you want to sum all columns, use the form ALL=S.
- You can specify column widths and compute column totals with one command line. When you want to perform both operations, list the compute column total (=S) after the width specification (=w).
- Example** `SELECT prodid prodprce=10=S proddesc=10 prodname FROM parts`
- This command displays the values of the columns *prodid*, *prodprce*, *proddesc*, and *prodname* in the table *parts*. Displays for *prodprce* and *proddesc* are ten characters wide, and the *prodprce* column total is displayed.